

Kai-Min Chung  
Yu Sasaki (Eds.)

LNCS 15486

# Advances in Cryptology – ASIACRYPT 2024

30th International Conference on the Theory  
and Application of Cryptology and Information Security  
Kolkata, India, December 9–13, 2024  
Proceedings. Part III

3  
Part III



 Springer

# Lecture Notes in Computer Science

15486


## Founding Editors


Gerhard Goos  
Juris Hartmanis

## Editorial Board Members

Elisa Bertino, *Purdue University, West Lafayette, IN, USA*

Wen Gao, *Peking University, Beijing, China*

Bernhard Steffen , *TU Dortmund University, Dortmund, Germany*

Moti Yung , *Columbia University, New York, NY, USA*

The series Lecture Notes in Computer Science (LNCS), including its subseries Lecture Notes in Artificial Intelligence (LNAI) and Lecture Notes in Bioinformatics (LNBI), has established itself as a medium for the publication of new developments in computer science and information technology research, teaching, and education.


LNCS enjoys close cooperation with the computer science R & D community, the series counts many renowned academics among its volume editors and paper authors, and collaborates with prestigious societies. Its mission is to serve this international community by providing an invaluable service, mainly focused on the publication of conference and workshop proceedings and postproceedings. LNCS commenced publication in 1973.


Kai-Min Chung · Yu Sasaki  
Editors

# Advances in Cryptology – ASIACRYPT 2024

30th International Conference on the Theory  
and Application of Cryptology and Information Security  
Kolkata, India, December 9–13, 2024  
Proceedings, Part III

*Editors*

Kai-Min Chung   
Academia Sinica  
Taipei, Taiwan

Yu Sasaki   
NTT Social Informatics Laboratories  
Tokyo, Japan

ISSN 0302-9743

ISSN 1611-3349 (electronic)

Lecture Notes in Computer Science

ISBN 978-981-96-0890-4

ISBN 978-981-96-0891-1 (eBook)

<https://doi.org/10.1007/978-981-96-0891-1>

© International Association for Cryptologic Research 2025

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Singapore Pte Ltd.  
The registered company address is: 152 Beach Road, #21-01/04 Gateway East, Singapore 189721, Singapore

If disposing of this product, please recycle the paper.

# Preface

The 30th Annual International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt 2024) was held in Kolkata, India, on December 9–13, 2024. The conference covered all technical aspects of cryptology and was sponsored by the International Association for Cryptologic Research (IACR).

We received a record 433 paper submissions for Asiacrypt from around the world. The Program Committee (PC) selected 127 papers for publication in the proceedings of the conference. As in the previous year, the Asiacrypt 2024 program had three tracks.

The two program chairs are greatly indebted to the six area chairs for their great contributions throughout the paper selection process. The area chairs were Siyao Guo for Information-Theoretic and Complexity-Theoretic Cryptography, Bo-Yin Yang for Efficient and Secure Implementations, Goichiro Hanaoka for Public-Key Cryptography Algorithms and Protocols, Arpita Patra for Multi-Party Computation and Zero-Knowledge, Prabhanjan Ananth for Public-Key Primitives with Advanced Functionalities, and Tetsu Iwata for Symmetric-Key Cryptography. The area chairs helped suggest candidates to form a strong program committee, foster and moderate discussions together with the PC members assigned as paper discussion leads to form consensus, suggest decisions on submissions in their areas, and nominate outstanding PC members. We are sincerely grateful for the invaluable contributions of the area chairs.

To review and evaluate the submissions, while keeping the load per PC member manageable, we selected the PC members consisting of 105 leading experts from all over the world, in all six topic areas of cryptology, and we also had approximately 468 external reviewers, whose input was critical to the selection of papers. The review process was conducted using double-blind peer review. The conference operated a two-round review system with a rebuttal phase. This year, we continued the interactive rebuttal from Asiacrypt 2023. After the reviews and first-round discussions, PC members and area chairs selected 264 submissions to proceed to the second round. The remaining 169 papers were rejected, including two desk-rejects. Then, the authors were invited to participate in a two-step interactive rebuttal phase, where the authors needed to submit a rebuttal in five days and then interact with the reviewers to address questions and concerns the following week. We believe the interactive form of the rebuttal encouraged discussions between the authors and the reviewers to clarify the concerns and contributions of the submissions and improved the review process. Then, after several weeks of second-round discussions, the committee selected the final 127 papers to appear in these proceedings. This year, we received seven resubmissions from the revise-and-resubmit experiment from Crypto 2024, of which five were accepted. The nine volumes of the conference proceedings contain the revised versions of the 127 papers that were selected. The final revised versions of papers were not reviewed again and the authors are responsible for their contents.

The PC nominated and voted for three papers to receive the Best Paper Awards. The Best Paper Awards went to Mariya Georgieva Belorgey, Sergiu Carпов, Nicolas Gama,

Sandra Guasch and Dimitar Jetchev for their paper “Revisiting Key Decomposition Techniques for FHE: Simpler, Faster and More Generic”, Xiaoyang Dong, Yingxin Li, Fukang Liu, Siwei Sun and Gaoli Wang for their paper “The First Practical Collision for 31-Step SHA-256”, and Valerio Cini and Hoeteck Wee for their paper “Unbounded ABE for Circuits from LWE, Revisited”. The authors of those three papers were invited to submit extended versions of their papers to the Journal of Cryptology.

The program of Asiacrypt 2024 also featured the 2024 IACR Distinguished Lecture delivered by Paul Kocher and one invited talk, nominated and voted by the PC. The invited speaker had not yet been determined when this preface was written. Following Eurocrypt 2024, we selected seven PC members for the Distinguished PC Members Awards, nominated by the area chairs and program chairs. The Outstanding PC Members Awards went to Sherman S. M. Chow, Elizabeth Crites, Matthias J. Kannwischer, Mustafa Khairallah, Ruben Niederhagen, Maciej Obremski and Keita Xagawa.

Following Crypto 2024, Asiacrypt 2024 included an artifact evaluation process for the first time. Authors of accepted papers were invited to submit associated artifacts, such as software or datasets, for archiving alongside their papers; 14 artifacts were submitted. Rei Ueno was the Artifact Chair and led an artifact evaluation committee of 10 members listed below. In the interactive review process between authors and reviewers, the goal was not just to evaluate artifacts but also to improve them. Artifacts that passed successfully through the artifact review process were publicly archived by the IACR at <https://artifacts.iacr.org/>.

Numerous people contributed to the success of Asiacrypt 2024. We would like to thank all the authors, including those whose submissions were not accepted, for submitting their research results to the conference. We are very grateful to the area chairs, PC members, and external reviewers for contributing their knowledge and expertise, and for the tremendous amount of work that was done with reading papers and contributing to the discussions. We are greatly indebted to Bimal Kumar Roy, the General Chairs, for their efforts in organizing the event, to Kevin McCurley and Kay McKelly for their help with the website and review system, and to Jih-Wei Shih for the assistance with the use of the review system. We thank the Asiacrypt 2024 advisory committee members Bart Preneel, Huaxiong Wang, Bo-Yin Yang, Goichiro Hanaoka, Jian Guo, Ron Steinfeld, and Michel Abdalla for their valuable suggestions. We are also grateful for the helpful advice and organizational material provided to us by Crypto 2024 PC co-chairs Leonid Reyzin and Douglas Stebila, Eurocrypt 2024 PC co-chairs Marc Joye and Gregor Leander, and TCC 2023 chair Hoeteck Wee. We also thank the team at Springer for handling the publication of these conference proceedings.

December 2024

Kai-Min Chung  
Yu Sasaki

# Organization

## General Chair

Bimal Kumar Roy

TCG CREST Kolkata, India

## Program Committee Chairs

Kai-Min Chung

Academia Sinica, Taiwan

Yu Sasaki

NTT Social Informatics Laboratories Tokyo,  
Japan and National Institute of Standards and  
Technology, USA

## Area Chairs

Prabhanjan Ananth

University of California, Santa Barbara, USA

Siyao Guo

NYU Shanghai, China

Goichiro Hanaoka

National Institute of Advanced Industrial Science  
and Technology, Japan

Tetsu Iwata

Nagoya University, Japan

Arpita Patra

Indian Institute of Science Bangalore, India

Bo-Yin Yang

Academia Sinica, Taiwan

## Program Committee

Akshima

NYU Shanghai, China

Bar Alon

Ben-Gurion University, Israel

Elena Andreeva

TU Wien, Austria

Nuttapong Attrapadung

AIST, Japan

Subhadeep Banik

University of Lugano, Switzerland

Zhenzhen Bao

Tsinghua University, China

James Bartusek

University of California, Berkeley, USA

Hanno Becker

Amazon Web Services, UK

Sonia Belaïd

CryptoExperts, France

Ward Beullens

IBM Research, Switzerland

Andrej Bogdanov

University of Ottawa, Canada



Pedro Branco	Max Planck Institute for Security and Privacy, Germany
Gaëtan Cassiers	UCLouvain, Belgium
Céline Chevalier	CRED, Université Paris-Panthéon-Assas, and DIENS, France
Avik Chakraborti	Institute for Advancing Intelligence TCG CREST, India
Nishanth Chandran	Microsoft Research India, India
Jie Chen	East China Normal University, China
Yu Long Chen	KU Leuven and National Institute of Standards and Technology, Belgium
Mahdi Cheraghchi	University of Michigan, USA
Nai-Hui Chia	Rice University, USA
Wonseok Choi	Purdue University, USA
Tung Chou	Academia Sinica, Taiwan
Arka Rai Choudhuri	NTT Research, USA
Sherman S. M. Chow	Chinese University of Hong Kong, China
Chitchanok Chuengsatiansup	University of Melbourne, Australia
Michele Ciampi	University of Edinburgh, UK
Valerio Cini	NTT Research, USA
Elizabeth Crites	Web3 Foundation, Switzerland
Nico Döttling	CISPA Helmholtz Center, Germany
Avijit Dutta	Institute for Advancing Intelligence TCG CREST, India
Daniel Escudero	JP Morgan AlgoCRYPT CoE and JP Morgan AI Research, USA
Thomas Espitau	PQShield, France
Jun Furukawa	NEC Corporation, Japan
Rosario Gennaro	CUNY, USA
Junqing Gong	East China Normal University, China
Rishab Goyal	University of Wisconsin-Madison, USA
Julia Hesse	IBM Research Europe, Switzerland
Akinori Hosoyamada	NTT Social Informatics Laboratories, Japan
Michael Hutter	PQShield, Austria
Takanori Isobe	University of Hyogo, Japan
Joseph Jaeger	Georgia Institute of Technology, USA
Matthias J. Kannwischer	Chelpis Quantum Corp, Taiwan
Bhavana Kanukurthi	Indian Institute of Science, India
Shuichi Katsumata	PQShield and AIST, Japan
Jonathan Katz	Google and University of Maryland, USA
Mustafa Khairallah	Lund University, Sweden
Fuyuki Kitagawa	NTT Social Informatics Laboratories, Japan

Karen Klein	ETH Zurich, Switzerland
Mukul Kulkarni	Technology Innovation Institute, United Arab Emirates
Po-Chun Kuo	WisdomRoot Tech, Taiwan
Jooyoung Lee	KAIST, South Korea
Wei-Kai Lin	University of Virginia, USA
Feng-Hao Liu	Washington State University, USA
Jiahui Liu	Massachusetts Institute of Technology, USA
Qipeng Liu	UC San Diego, USA
Shengli Liu	Shanghai Jiao Tong University, China
Chen-Da Liu-Zhang	Lucerne University of Applied Sciences and Arts and Web3 Foundation, Switzerland
Yun Lu	University of Victoria, Canada
Ji Luo	University of Washington, USA
Silvia Mella	Radboud University, Netherlands
Peihan Miao	Brown University, USA
Daniele Micciancio	UCSD, USA
Yusuke Naito	Mitsubishi Electric Corporation, Japan
Khoa Nguyen	University of Wollongong, Australia
Ruben Niederhagen	Academia Sinica, Taiwan and University of Southern Denmark, Denmark
Maciej Obremski	National University of Singapore, Singapore
Miyako Ohkubo	NICT, Japan
Eran Omri	Ariel University, Israel
Jiaxin Pan	University of Kassel, Germany
Anat Paskin-Cherniavsky	Ariel University, Israel
Goutam Paul	Indian Statistical Institute, India
Chris Peikert	University of Michigan, USA
Christophe Petit	University of Birmingham and Université libre de Bruxelles, Belgium
Rachel Player	Royal Holloway University of London, UK
Thomas Prest	PQShield, France
Shahram Rasoolzadeh	Ruhr University Bochum, Germany
Alexander Russell	University of Connecticut, USA
Santanu Sarkar	IIT Madras, India
Sven Schäge	Eindhoven University of Technology, Netherlands
Gregor Seiler	IBM Research Europe, Switzerland
Sruthi Sekar	Indian Institute of Technology, India
Yaobin Shen	Xiamen University, China
Danping Shi	Institute of Information Engineering, Chinese Academy of Sciences, China
Yifan Song	Tsinghua University, China

Katerina Sotiraki	Yale University, USA
Akshayaram Srinivasan	University of Toronto, Canada
Marc Stöttinger	Hochschule RheinMain, Germany
Akira Takahashi	J.P. Morgan AI Research and AlgoCRYPT CoE, USA
Qiang Tang	University of Sydney, Australia
Aishwarya Thiruvengadam	IIT Madras, India
Emmanuel Thomé	Inria Nancy, France
Junichi Tomida	NTT Social Informatics Laboratories, Japan
Monika Trimoska	Eindhoven University of Technology, Netherlands
Huaxiong Wang	Nanyang Technological University, Singapore
Meiqin Wang	Shandong University, China
Qingju Wang	Telecom Paris, Institut Polytechnique de Paris, France
David Wu	UT Austin, USA
Keita Xagawa	Technology Innovation Institute, United Arab Emirates
Chaoping Xing	Shanghai Jiaotong University, China
Shiyuan Xu	University of Hong Kong, China
Anshu Yadav	IST, Austria
Shota Yamada	AIST, Japan
Yu Yu	Shanghai Jiao Tong University, China
Mark Zhandry	NTT Research, USA
Hong-Sheng Zhou	Virginia Commonwealth University, USA

## Additional Reviewers

Hugo Aaronson	Jiawei Bao
Damiano Abram	Jyotirmoy Basak
Hamza Abusalah	Nirupam Basak
Abtin Afshar	Gabrielle Beck
Siddharth Agarwal	Hugo Beguinet
Navid Alapati	Amit Behera
Miguel Ambrona	Mihir Bellare
Parisa Amiri Eliasi	Tamar Ben David
Ravi Anand	Aner Moshe Ben Efraim
Saikrishna Badrinarayanan	Fabrice Benhamouda
Chen Bai	Tyler Besselman
David Balbás	Tim Beyne
Brieuc Balon	Rishabh Bhadauria
Gustavo Banegas	Divyanshu Bhardwaj
Laasya Bangalore	Shivam Bhasin

Amit Singh Bhati  
Loïc Bidoux  
Alexander Bienstock  
Jan Bobolz  
Alexandra Boldyreva  
Maxime Bombar  
Nicolas Bon  
Carl Bootland  
Jonathan Bootle  
Giacomo Borin  
Cecilia Boschini  
Jean-Philippe Bossuat  
Mariana Botelho da Gama  
Christina Boura  
Pierre Briaud  
Jeffrey Burdges  
Fabio Campos  
Yibo Cao  
Pedro Capitão  
Ignacio Cascudo  
David Cash  
Wouter Castryck  
Anirban Chakrabartha  
Debasmita Chakraborty  
Suvradip Chakraborty  
Kanad Chakravarti  
Ayantika Chatterjee  
Rohit Chatterjee  
Jorge Chavez-Saab  
Binyi Chen  
Bohang Chen  
Long Chen  
Mingjie Chen  
Shiyao Chen  
Xue Chen  
Yu-Chi Chen  
Chen-Mou Cheng  
Jiaqi Cheng  
Ashish Choudhury  
Miranda Christ  
Qiaohan Chu  
Eldon Chung  
Hao Chung  
Léo Colisson  
Daniel Collins  
Jolijn Cottaar  
Murilo Coutinho  
Eric Crockett  
Bibhas Chandra Das  
Nayana Das  
Pratish Datta  
Alex Davidson  
Hannah Davis  
Leo de Castro  
Luca De Feo  
Thomas Decru  
Giovanni Deligios  
Ning Ding  
Fangqi Dong  
Minxin Du  
Qiuyan Du  
Jesko Dujmovic  
Moumita Dutta  
Pranjal Dutta  
Duyen  
Marius Eggert  
Solane El Hirsch  
Andre Esser  
Hülya Evkan  
Sebastian Faller  
Yanchen Fan  
Niklas Fassbender  
Hanwen Feng  
Xiutao Feng  
Dario Fiore  
Scott Fluhrer  
Danilo Francati  
Shiuan Fu  
Georg Fuchsbauer  
Shang Gao  
Rachit Garg  
Gayathri Garimella  
Pierrick Gaudry  
François Gérard  
Paul Gerhart  
Riddhi Ghosal  
Shibam Ghosh  
Ashrujit Ghoshal  
Shane Gibbons  
Valerie Gilchrist

Xinxin Gong  
Lorenzo Grassi  
Scott Griffy  
Chaowen Guan  
Aurore Guillevic  
Sam Gunn  
Felix Günther  
Kanav Gupta  
Shreyas Gupta  
Kamil Doruk Gur  
Jincheol Ha  
Hossein Hadipour  
Tovohery Hajatiana Randrianarisoa  
Shai Halevi  
Shuai Han  
Tobias Handirk  
Yonglin Hao  
Zihan Hao  
Keisuke Hara  
Keitaro Hashimoto  
Aditya Hegde  
Andreas Hellenbrand  
Paul Hermouet  
Minki Hhan  
Hilder Lima  
Taiga Hiroka  
Ryo Hiromasa  
Viet Tung Hoang  
Charlotte Hoffmann  
Clément Hoffmann  
Man Hou Hong  
Wei-Chih Hong  
Alexander Hoover  
Fumitaka Hoshino  
Patrick Hough  
Yao-Ching Hsieh  
Chengcong Hu  
David Hu  
Kai Hu  
Zihan Hu  
Hai Huang  
Mi-Ying Huang  
Yu-Hsuan Huang  
Zhicong Huang  
Shih-Han Hung  
Yuval Ishai  
Ryoma Ito  
Amit Jana  
Ashwin Jha  
Xiaoyu Ji  
Yanxue Jia  
Mingming Jiang  
Lin Jiao  
Haoxiang Jin  
Zhengzhong Jin  
Chris Jones  
Eliran Kachlon  
Giannis Kaklamanis  
Chethan Kamath  
Soumya Kanti Saha  
Sabyasachi Karati  
Harish Karthikeyan  
Andes Y. L. Kei  
Jean Kieffer  
Jiseung Kim  
Seongkwang Kim  
Sebastian Kolby  
Sreehari Kollath  
Dimitris Kolonelos  
Venkata Koppula  
Abhiram Kothapalli  
Stanislav Kruglik  
Anup Kumar Kundu  
Péter Kutas  
Norman Lahr  
Qiqi Lai  
Yi-Fu Lai  
Abel Laval  
Guirec Lebrun  
Byeonghak Lee  
Changmin Lee  
Hyung Tae Lee  
Joohee Lee  
Keewoo Lee  
Yeongmin Lee  
Yongwoo Lee  
Andrea Lesavourey  
Baiyu Li  
Jiangtao Li  
Jianqiang Li

Junru Li  
Liran Li  
Minzhang Li  
Shun Li  
Songsong Li  
Weihan Li  
Wenzhong Li  
Yamin Li  
Yanan Li  
Yu Li  
Yun Li  
Zeyong Li  
Zhe Li  
Chuanwei Lin  
Fuchun Lin  
Yao-Ting Lin  
Yunhao Ling  
Eik List  
Fengrun Liu  
Fukang Liu  
Hanlin Liu  
Hongqing Liu  
Rui Liu  
Tianren Liu  
Xiang Liu  
Xiangyu Liu  
Zeyu Liu  
Paul Lou  
George Lu  
Zhenghao Lu  
Ting-Gian Lua  
You Lyu  
Jack P. K. Ma  
Yiping Ma  
Varun Madathil  
Lorenzo Magliocco  
Avishek Majumder  
Nikolaos Makriyannis  
Varun Maram  
Chloe Martindale  
Elisaweta Masserova  
Jake Massimo  
Loïc Masure  
Takahiro Matsuda  
Christian Matt  
Subhra Mazumdar  
Nikolas Melissaris  
Michael Meyer  
Ankit Kumar Misra  
Anuja Modi  
Deep Inder Mohan  
Charles Momin  
Johannes Mono  
Hart Montgomery  
Ethan Mook  
Thorben Moos  
Tomoyuki Morimae  
Hiraku Morita  
Tomoki Moriya  
Aditya Morolia  
Christian Mouchet  
Nicky Mouha  
Tamer Mour  
Changrui Mu  
Arindam Mukherjee  
Pratyay Mukherjee  
Anne Müller  
Alice Murphy  
Shyam Murthy  
Kohei Nakagawa  
Barak Nehoran  
Patrick Neumann  
Lucien K. L. Ng  
Duy Nguyen  
Ky Nguyen  
Olga Nissenbaum  
Anca Nitulescu  
Julian Nowakowski  
Frederique Oggier  
Jean-Baptiste Orfila  
Emmanuela Orsini  
Tapas Pal  
Ying-yu Pan  
Roberto Parisella  
Aditi Partap  
Alain Passelègue  
Alice Pellet-Mary  
Zachary Pepin  
Octavio Perez Kempner  
Edoardo Perichetti

Léo Perrin  
Naty Peter  
Richard Petri  
Rafael del Pino  
Federico Pintore  
Erik Pohle  
Simon Pohmann  
Guru Vamsi Policharla  
Daniel Pollman  
Yuriy Polyakov  
Alexander Poremba  
Eamonn Postlethwaite  
Sihang Pu  
Luowen Qian  
Tian Qiu  
Rajeev Raghunath  
Srinivasan Raghuraman  
Mostafizar Rahman  
Mahesh Rajasree  
Somindu Chaya Ramanna  
Simon Rastikian  
Anik Raychaudhuri  
Martin Rehberg  
Michael Reichle  
Krijn Reijnders  
Doreen Riepel  
Guilherme Rito  
Matthieu Rivain  
Bhaskar Roberts  
Marc Roeschlin  
Michael Rosenberg  
Paul Rösler  
Arnab Roy  
Lawrence Roy  
Luigi Russo  
Keegan Ryan  
Markku-Juhani Saarinen  
Éric Sageloli  
Dhiman Saha  
Sayandeep Saha  
Yusuke Sakai  
Kosei Sakamoto  
Subhabrata Samajder  
Simona Samardjiska  
Maria Corte-Real Santos  
Sina Schaeffler  
André Schrottenloher  
Jacob Schuldt  
Mark Schultz  
Mahdi Sedaghat  
Jae Hong Seo  
Yannick Seurin  
Aein Shahmirzadi  
Girisha Shankar  
Yixin Shen  
Rentaro Shiba  
Ardeshir Shojaeinasab  
Jun Jie Sim  
Mark Simkin  
Jaspal Singh  
Benjamin Smith  
Yongha Son  
Fang Song  
Yongsoo Song  
Pratik Soni  
Pierre-Jean Spaenlehauer  
Matthias Johann Steiner  
Lukas Stennes  
Roy Stracovsky  
Takeshi Sugawara  
Adam Suhl  
Siwei Sun  
Elias Suvanto  
Koutarou Suzuki  
Erkan Tairi  
Atsushi Takayasu  
Kaoru Takemure  
Abdullah Talayhan  
Quan Quan Tan  
Gang Tang  
Khai Hanh Tang  
Tianxin Tang  
Yi Tang  
Stefano Tessaro  
Sri AravindaKrishnan Thyagarajan  
Yan Bo Ti  
Jean-Pierre Tillich  
Toi Tomita  
Aleksei Udoenko  
Arunachalaeswaran V.

Aron van Baarsen	Kyosuke Yamashita
Wessel van Woerden	Jiayun Yan
Michiel Verbauwhede	Yingfei Yan
Corentin Verhamme	Qianqian Yang
Quoc-Huy Vu	Rupeng Yang
Benedikt Wagner	Xinrui Yang
Julian Wälde	Yibin Yang
Hendrik Waldner	Zhaomin Yang
Judy Walker	Yizhou Yao
Alexandre Wallet	Kevin Yeo
Han Wang	Eylon Yogev
Haoyang Wang	Yusuke Yoshida
Jiabo Wang	Aaram Yun
Jiafan Wang	Gabriel Zaid
Liping Wang	Riccardo Zanotto
Mingyuan Wang	Shang Zehua
Peng Wang	Hadas Zeilberger
Weihao Wang	Runzhi Zeng
Yunhao Wang	Bin Zhang
Zhedong Wang	Cong Zhang
Yohei Watanabe	Liu Zhang
Chenkai Weng	Tianwei Zhang
Andreas Weninger	Tianyu Zhang
Stella Wohnig	Xiangyang Zhang
Harry W. H. Wong	Yijian Zhang
Ivy K. Y. Woo	Yinuo Zhang
Tiger Wu	Yuxin Zhang
Yu Xia	Chang-an Zhao
Zejun Xiang	Tianyu Zhao
Yuting Xiao	Yu Zhou
Ning Xie	Yunxiao Zhou
Zhiye Xie	Zhelei Zhou
Lei Xu	Zibo Zhou
Yanhong Xu	Chenzhi Zhu
Haiyang Xue	Ziqi Zhu
Aayush Yadav	Cong Zuo
Saikumar Yadugiri	

## Artifact Chair

Rei Ueno

Kyoto University, Japan



## Artifact Evaluation Committee

Julien Béguinot	LTCl, Télécom Paris, Institut Polytechnique de Paris, France
Aron Gohr	Independent Researcher
Hosein Hadipour	Graz University of Technology, Austria
Akira Ito	NTT Social Informatics Laboratories, Japan
Haruto Kimura	University of Melbourne, Australia and Waseda University, Japan
Kotaro Matsuoka	Kyoto University, Japan
Florian Mendel	Infineon Technologies, Germany
Hiraku Morita	Aarhus University, University of Copenhagen, Denmark
Prasanna Ravi	Nanyang Technological University, Singapore
Élise Tasso	Tohoku University, Japan

## Contents – Part III

### Pairing-Based Cryptography

Non-malleable Subvector Commitments .....	3
<i>Benoît Libert</i>	

Traitor Tracing Without Trusted Authority from Registered Functional Encryption .....	33
<i>Pedro Branco, Russell W. F. Lai, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Ivy K. Y. Woo</i>	

### Threshold Cryptography

Interactive Threshold Mercurial Signatures and Applications .....	69
<i>Masayuki Abe, Masaya Nanri, Octavio Perez Kempner, and Mehdi Tibouchi</i>	

HARTS: High-Threshold, Adaptively Secure, and Robust Threshold Schnorr Signatures .....	104
<i>Renas Bacho, Julian Loss, Gilad Stern, and Benedikt Wagner</i>	

Tiresias: Large Scale, UC-Secure Threshold Paillier .....	141
<i>Offir Friedman, Avichai Marmor, Dolev Mutzari, Yehonatan C. Scalay, Yuval Spiizer, and Avishay Yanai</i>	

Password-Protected Threshold Signatures .....	174
<i>Stefan Dziembowski, Stanislaw Jarecki, Pawel Kedzior, Hugo Krawczyk, Chan Nam Ngo, and Jiayu Xu</i>	

Strongly Secure Universal Thresholdizer .....	207
<i>Ehsan Ebrahimi and Anshu Yadav</i>	

### Isogeny-Based Cryptography

Ideal-to-Isogeny Algorithm Using 2-Dimensional Isogenies and Its Application to SQIsign .....	243
<i>Hiroshi Onuki and Kohei Nakagawa</i>	

<b>SQISign2D-East: A New Signature Scheme Using 2-Dimensional Isogenies . . .</b>	<b>272</b>
<i>Kohei Nakagawa, Hiroshi Onuki, Wouter Castryck, Mingjie Chen, Riccardo Invernizzi, Gioella Lorenzon, and Frederik Vercauteren</i>	
<b>An Algorithmic Approach to <math>(2, 2)</math>-Isogenies in the Theta Model and Applications to Isogeny-Based Cryptography . . . . .</b>	<b>304</b>
<i>Pierrick Dartois, Luciano Maino, Giacomo Pope, and Damien Robert</i>	
<b>SQISign2D–West: The Fast, the Small, and the Safer . . . . .</b>	<b>339</b>
<i>Andrea Basso, Pierrick Dartois, Luca De Feo, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski</i>	
<b>Extending Class Group Action Attacks via Sesquilinear Pairings . . . . .</b>	<b>371</b>
<i>Joseph Macula and Katherine E. Stange</i>	
<b>SQIPrime: A Dimension 2 Variant of SQISignHD with Non-smooth Challenge Isogenies . . . . .</b>	<b>396</b>
<i>Max Duparc and Tako Boris Fouotsa</i>	
<b>Post-quantum Cryptography</b>	
<b>CPA-Secure KEMs are also Sufficient for Post-quantum TLS 1.3 . . . . .</b>	<b>433</b>
<i>Biming Zhou, Haodong Jiang, and Yunlei Zhao</i>	
<b>Post-quantum Asynchronous Remote Key Generation for FIDO2 . . . . .</b>	<b>465</b>
<i>Jacqueline Brendel, Sebastian Clermont, and Marc Fischlin</i>	
<b>Author Index . . . . .</b>	<b>495</b>

# **Pairing-Based Cryptography**



# Non-malleable Subvector Commitments

Benoît Libert<sup>(✉)</sup>

Zama, Paris, France

`benoit.libert@zama.ai`

**Abstract.** Vector commitments are compressing commitments to vectors allowing for short local openings. Rotem and Segev (TCC'21) formalized a notion of non-malleability for vector commitments, which accounts for the information revealed by local openings when an adversary outputs its own commitment and attempts to open it to messages related to those of honest parties. They left open the problem of extending their non-malleable construction to the scenario of subvector commitments, where a committer can compactly open a significant fraction of committed vectors. In this paper, we construct non-malleable subvector commitments by generalizing Garay *et al.*'s notion of tag-based simulation-sound trapdoor commitments (Eurocrypt'03) to the subvector commitment setting. We then construct simulation-sound subvector commitments from the Bilinear Diffie-Hellman assumption as well as the Strong RSA and Bilinear Strong Diffie-Hellman assumptions. These constructions allow the adversary to see equivocations on multiple tags, and thus yield reusable (as defined by Damgård and Groth) non-malleable commitments.

**Keywords:** Vector commitments · subvector openings · non-malleability

## 1 Introduction

Vector commitments (VCs) [14, 46] allow one to commit to a vector  $\mathbf{m}$  by means of a short commitment string. Later on, the committer can locally open each component of  $\mathbf{m} = (m_1, \dots, m_n)$  by revealing a short opening. Importantly, both the commitment and its position-wise openings should have sub-linear (ideally, constant) size in the vector dimension  $n$ .

Like ordinary commitments, a vector commitment scheme should be *binding* (i.e., no efficient adversary can open a commitment to two distinct messages  $m_i, m'_i$  at the same position  $i \in [n]$ ) and *hiding* (meaning that commitments to distinct vectors should be indistinguishable).

Vector commitments have found a number of applications in zero-knowledge databases [46], verifiable data streaming [43], authenticated dictionaries [45, 64], de-centralized storage [12], succinct arguments [8, 44], cryptocurrencies [63] and stateless blockchains [8, 37].

Subvector commitments (SVCs) [8, 44] extend vector commitments by allowing the committer to open an entire subvector via a sublinear or even constant-size opening, no matter how large the committed vector or the opened subvector are. The compactness of subvector openings is useful in all applications of VCs

where many positions can be opened for a given commitment. In particular, it allows reducing communication costs in PCP/PIOP-based succinct arguments [8, 44], keyless proofs of retrievability [25], and decentralized storage [12]. In zero-knowledge databases [55], it can also enable further bandwidth savings [16] when multiple proofs are generated for database entries in the same region.

Number-theoretic constructions [8, 12, 44] of (sub)vector commitments often provide constant-size openings. On the downside, their homomorphic property (while necessary to support public updates [14]) comes with the potential threat of malleability attacks. As mentioned by Rotem and Segev [58], this malleability may become problematic when VCs are used in the context of blockchain transactions or multi-round sealed-bid auctions.

For example, in blockchain transactions and smart contracts, vector commitments can be stored by validators and contain compressed versions of a current state (e.g., account balances or user-specific properties). If the VC scheme in use is malleable, malicious users may be able to observe commitments and their local openings and maul them into a VC commitment to a related state (thus preventing transaction non-malleability [4]) with malleated local openings. In simultaneous multi-round sealed-bid auctions [5], bidders can bid for multiple items at each round. If an item is not sold at some round, a malicious bidder can attempt to maul honest bidders' commitments and local openings at this round so as to overbid in the next round.

Rotem and Segev [58] formalized a notion of non-malleability for vector commitments along the lines of non-interactive non-malleable commitments [20, 21]. They gave a generic construction of non-malleable vector commitment from ordinary VCs and a primitive called all-but-one binding locally equivocal commitment. Rotem and Segev [58] left open the problem of realizing their notion of non-malleability in subvector commitments. In this paper, we address this question and provide several constructions of non-malleable SVCs based on long-lived hardness assumptions. We prove them non-malleable in the standard model in the sense of a definition that extends the one of [58] to the SVC setting.

We note that subvector openings (in particular those obtained by aggregating individual openings) may be desirable in the applications suggested in [58]. In sealed-bid auctions, when bidders want to bid for many items, they can send the auctioneer a subvector opening to these items in order to save space. Then, when an individual winning bid is to be publicly revealed, the bidder can send the initial (pre-aggregation) local openings. In the blockchain/cryptocurrencies use case, (aggregatable) subvector openings were considered in [8, 37, 63]. Our schemes can be used to compress the local openings describing the state of an account memory, as suggested by PointProofs [37, Section 5]. In [37, Figure 3], it is assumed that accounts have around 1000 memory locations and up to 256 of them can be simultaneously opened. For such values, even subvector openings or same-commitment aggregation can lead to substantial savings.

**OUR CONTRIBUTIONS.** We provide three constructions of non-malleable subvector commitments based on specific number theoretic problems. The first one relies on the bilinear Diffie-Hellman assumption [9] (note that discrete-logarithm-

based VCs in the standard model can hardly avoid the use of pairings, as shown in [15]). This construction is a derivative of the Diffie-Hellman-based VC put forth by Catalano and Fiore [14] and performs about as well. In particular, it inherits its quadratic-size common reference string (CRS).

We give a second SVC based on the Strong RSA assumption [2] where the CRS size is constant. This construction actually trades an  $O(n^2)$ -size CRS for an asymptotically slower commitment algorithm computing  $O(n \cdot \log n)$  exponentiations. This scheme has a natural analogue based on the Bilinear Strong Diffie-Hellman assumption [7] with an  $O(n)$ -size CRS and where the committer computes  $O(n \cdot \log^3 n)$  field operations but only  $O(n)$  exponentiations.

Our constructions based on BDH and Strong RSA (or Bilinear Strong Diffie-Hellman) enable re-usability [18] in the sense that they still guarantee non-malleability in an experiment where the adversary is given a polynomial number of honestly generated commitments (while the definition and the construction of [58] only allow the adversary to obtain one honestly generated commitment).

At the expense of giving up re-usability, we provide a third, more efficient realization based on the standard RSA assumption. In this construction, the CRS and the commitment time are both strictly linear in the vector dimension. We leave it as an interesting open problem to simultaneously obtain a linear-time commitment phase and a CRS of size at most  $O(n)$  while retaining reusability.

**TECHNICAL OVERVIEW.** We first explain why the technique of Rotem and Segev [58] does not extend to subvector commitments. As pointed out in [58, Section 3.2], non-malleable VCs cannot be generically built by sequentially composing an ordinary VC and a standard non-malleable commitment (in this order) in a straightforward way. The reason is that, in their definition, the adversary is allowed to see local openings (which would contain an inner, possibly malleable vector commitment) of the honestly generated VC before outputting its own commitment. Instead, the generic construction of [58] sequentially combines an inner non-interactive tag-based commitment and an outer standard vector commitment. Each individual vector element  $m_i$  is committed to using a tag-based commitment labeled with a tag consisting of a one-time signature verification key. The obtained commitments  $(c_i)_{i \in [n]}$  are then compressed into a vector commitment which is finally signed using the one-time signature secret key.

Rotem and Segev [58] proved that the above construction works if the underlying tag-based commitment is locally equivocal and provides a property called all-but-one binding (its properties resemble Fischlin’s notion of identity-based trapdoor commitment [26]). This equivocability of the inner commitment implies that it must be randomized, meaning that each of the commitments  $(c_i)_{i \in [n]}$  has a non-trivial de-commitment information  $d_i$  that should be part of the local openings in the non-malleable VC. So, if we simply modify the construction of [58] and replace the outer VC by an SVC, the resulting construction is no longer compact: If the committer wants to open the first half  $(m_1, \dots, m_{n/2})$  of the committed vector  $\mathbf{m}$ , it has to reveal the de-commitments  $(d_i)_{i \in [n/2]}$  of the inner layer of locally equivocal commitments, thus ending up with  $O(n)$ -size local openings even if the outer VC supports compact subvector openings.

To avoid the above hurdle, we cannot hope to combine standard SVCs and standard tag-based commitments. Instead, we consider a notion of tag-based sometimes-binding equivocable subvector commitments. We actually extend the notion of simulation-sound trapdoor commitment (SSTC) [31, 52] into simulation-sound SVCs. We then show that the standard technique of building non-malleable commitments from simulation-sound ones extends to SVCs.

We note that we do not realize simulation-sound SVCs from any SVC. Informally, SSTCs are tag-based commitments where an adversary that observes equivocations of commitments for tags of its choice remains unable to equivocate commitments on a different tag. While standard SSTCs can be obtained from any signature [52] by using a  $\Sigma$ -protocol that allows proving knowledge of a signature,<sup>1</sup> this approach does not extend to commit to vectors while retaining short local openings. Although we cannot generically build simulation-sound SVCs, we can still realize them under fairly standard assumptions. Informally, our approach is to plug the verification mechanism of a signature scheme into the one of an SVC scheme in a non-generic (but still efficient) way.

Our BDH-based scheme is a non-malleable variant of a subvector commitment proposed by Gorbunov *et al.* [37], which is itself inspired by a construction of Lai and Malavolta [44] and obtained from the CDH-based VC of Catalano and Fiore [14]. In symmetric pairings  $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$  (which we assume in this overview for simplicity), the CRS contains group elements  $(g^{u_i}, g^{v_i})_{i=1}^n$  and  $(g^{u_i \cdot v_j})_{i \neq j}$ . A deterministic commitment to  $(m_1, \dots, m_n)$  is obtained as  $C = g^{\sum_{i=1}^n u_i \cdot m_i}$ . To open  $C$  to a subvector  $(m_i)_{i \in S}$ , the sender can use the CRS components  $(g^{u_i \cdot v_j})_{i \neq j}$  to compute  $\pi_S \in \mathbb{G}$  such that

$$e(C, g^{\sum_{j \in S} v_j}) = e(\pi_S, g) \cdot \prod_{i \in S} e(g^{u_i}, g^{v_i})^{m_i}. \quad (1)$$

Intuitively, the scheme is binding since  $(g^{u_i \cdot v_i})_{i=1}^n$  are not available in the CRS. As a result, when we expand the product  $(\sum_{i=1}^n u_i \cdot m_i) \cdot (\sum_{j \in S} v_j)$  in the left-hand-side member of (1), the terms  $(m_i \cdot u_i \cdot v_i)_{i \in S}$  are only computable in the exponent in  $\mathbb{G}_T$  (and not in  $\mathbb{G}$ ). If we randomize the commitment into  $C = g^{\gamma + \sum_{i=1}^n m_i \cdot u_i}$  for a random  $\gamma \in \mathbb{Z}_p$ , this suggests that a commitment  $C = g^\gamma$  to the zero-vector  $\mathbf{0}^n$  can be equivocated by using  $(g^{u_i \cdot v_i})_{i=1}^n$  to trapdoor-open  $C$  to any subvector  $(m_i)_{i \in [S]}$ . In order to obtain an SSTC, we replace the Diffie-Hellman values  $(g^{u_i \cdot v_i})_{i=1}^n$  by Waters signatures. A Waters signature [65] on a tag is a pair  $(\sigma_{i,1}, \sigma_{i,2}) = (g^{u_i \cdot v_i} \cdot H(\mathbf{tag})^r, g^r)$ , for some number-theoretic hash function  $H : \{0, 1\}^* \rightarrow \mathbb{G}$ , which satisfies a verification equation of the form  $e(\sigma_{i,1}, g) = e(g^{u_i}, g^{v_i}) \cdot e(H(\mathbf{tag}), \sigma_{i,2})$ . By exploiting the special shape of verification equations, we can fold the one of the signature scheme into (1) and

<sup>1</sup> A message  $m$  is committed to by using it as the challenge of a  $\Sigma$ -protocol in an HVZK-simulated execution of the  $\Sigma$ -protocol allowing to prove knowledge of a signature on the tag. The commitment is the simulated “first prover message” and its opening is the simulated response.



replace the local opening  $\pi_S$  by a pair  $(\pi_{S,1}, \pi_{S,2}) \in \mathbb{G}^2$  satisfying

$$e(C, g^{\sum_{j \in S} v_j}) = e(\pi_{S,1}, g) \cdot e(\pi_{S,2}, H(\text{tag})) \cdot \prod_{i \in S} e(g^{u_i}, g^{v_i})^{m_i}. \quad (2)$$

This allows equivocating  $(C, \text{tag})$  by using  $(g^{u_i \cdot v_i} \cdot H(\text{tag})^r, g^r)$  without knowing  $g^{u_i v_i}$  itself. A similar technique was used in [46] to build a multi-trapdoor mercurial [17] commitment.<sup>2</sup>

In the security proof, we cannot generically turn a simulation-soundness adversary into a signature forger.<sup>3</sup> Instead, we have to combine the proof techniques of Waters signatures and that of the subvector commitment of Gorbunov *et al.* (like [37], we need a stronger assumption than the CDH assumption implied by the unforgeability of Waters signatures).

Our construction from the Strong RSA assumption is based on the RSA-based VC of [14] and its subvector variant proposed by Lai and Malavolta [44]. In [14], the CRS contains a set of prime exponents  $\{e_i\}_{i=1}^n$  that do not divide  $\varphi(N)$  and a set of elements  $(g_0, g_1, \dots, g_n) \in (\mathbb{Z}_N^*)^n$  where  $g_i = g_0^{\prod_{j \in [n] \setminus \{i\}} e_j} \bmod N$  for each  $i \in [n]$ . A deterministic commitment to  $(m_1, \dots, m_n)$  is then obtained as  $C = \prod_{i=1}^n g_i^{m_i}$ . In order to open a subvector  $(m_i)_{i \in S}$ , the committer of [44] reveals an opening  $\pi_S = (C \cdot \prod_{i \in S} g_i^{-m_i})^{1/e_S} \bmod N$ ,<sup>4</sup> where  $e_S = \prod_{i \in S} e_i$ , which satisfies the verification equation  $C = \pi_S^{e_S} \cdot \prod_{i \in S} g_i^{m_i} \bmod N$ . To obtain an SSTC from this commitment, we first need to randomize it while preserving its local openability. A simple solution is to commit to  $(m_1, \dots, m_n)$  as  $C = g^\gamma \cdot \prod_{i=1}^n g_i^{m_i}$ , where  $g = g_0^{\prod_{i \in S} e_i} \bmod N$ , for a random  $\gamma \in \mathbb{Z}_{(N-1)/4}$ . We note that the discrete logarithm  $e_i = \log_{g_i}(g)$  is publicly known. However, it does not affect the binding property (what matters is that  $\log_g(g_i) = e_i^{-1} \bmod \text{ord}(g)$  be hidden). We next have to find a way to introduce tags in the scheme in a way that provides simulation-soundness. To do this, we adapt a technique used in [33], which uses Gennaro-Halevi-Rabin signatures [34] as equivocation trapdoors to equivocate Strong-RSA-based commitments.<sup>5</sup> Here, we apply the same technique in parallel to each  $g_i$  and derive the prime exponents  $\{e_i\}_{i=1}^n$  from tags by applying a hash function outputting prime numbers. For each tag  $\text{tag}$ , we derive a different set of primes  $\{e_{\text{tag},i} = H(\text{tag}, i)\}_{i=1}^n$  that defines a different tag-based commitment key. Since an  $e_i$ -th root of  $g_i$  can be used to equivocate the  $i$ -th position, the reduction programs the CRS so as to know an  $e_{\text{tag},i}$ -th root of  $g_i$  for each  $i \in [n]$  and each tag involved in equivocation queries. Under the Strong RSA assumption, we can prove that the scheme is unbounded simulation-sound binding under non-adaptive-tag queries, which is sufficient to construct non-malleable SVCs.

<sup>2</sup> The above scheme can actually be turned into a multi-trapdoor [33] SVC.

<sup>3</sup> One reason is that the binding adversary can output a commitment  $C$  and subvector openings  $\mathbf{m}[S_1] \neq \mathbf{m}[S_2]$  for two sets  $S_1, S_2$  that are not identical but just have a non-empty intersection. A direct reduction from the security of Waters signatures would be possible if we had  $S_1 = S_2$ .

<sup>4</sup> This is computable as  $\pi_S = g_0^{\sum_{i \in [n] \setminus S} m_i \cdot \prod_{j \in [n] \setminus (S \cup \{i\})} e_j} \bmod N$  without knowing the factorization.

<sup>5</sup> Namely, a commitment  $C = g^m \cdot r^{e_{\text{tag}}} \bmod N$  can be equivocated knowing an  $e_{\text{tag}}$ -th root of a fixed  $g$ .

In our RSA-based one-time-simulation-binding variant, we do not need a hash function that outputs prime numbers. Instead, we use a technique previously used by Di Crescenzo *et al.* [21] (and also in the number-theoretic all-but-one-binding equivocable commitments of Rotem and Segev [59, Appendix B.2]) to program the common reference string in such a way that the reduction can answer equivocation queries for exactly one tag. At the same time, an equivocation on any other tag reveals a solution to the given problem instance.

RELATED WORK. Subvector commitments were independently formalized by Lai and Malavolta [44] and Boneh *et al.* [8] who gave instantiations in hidden-order groups. Lai and Malavolta [44] also showed that a variant of the CDH-based VC of [14] allows subvector openings. Back in 2015, subvector commitments were implicitly described by Camenisch *et al.* [11, Section 3.1] who realized it from polynomial commitments with batch openings [42].

Vector commitments that support proof aggregation [8, 12, 37, 62, 63] immediately imply SVCs, by simply aggregating same-commitment proofs. Campanelli *et al.* [12] defined incrementally aggregatable VCs, where different subvector openings can be merged into a short opening for the union of their subvectors. Moreover, aggregated proofs support further aggregation. They realized incrementally aggregatable VCs in hidden-order groups.

Gorbunov *et al.* [37] extended the Libert-Yung VC [46] to enable cross-commitment aggregation. They also showed [37, Appendix A] that a variant of Catalano and Fiore’s CDH-based VC [14] supports same-commitment aggregation. Their construction is the basis of our first simulation-sound SVC scheme.

Non-malleable cryptography was first introduced in the seminal work of Dolev, Dwork and Naor [22, 23]. Since then a very large body of work (see, e.g., [10, 13, 18, 20, 21, 27, 28, 32, 33, 38–40, 47–50, 52, 57]) was devoted to the study of non-malleable commitments. In the context of vector commitments, the problem was addressed for the first time in the work of Gennaro and Micali [35] on independent zero-knowledge databases [55]. ZK-EDBs [55] can be seen as a generalized form of size-hiding vector commitments supporting non-membership proofs. They may be overkill for applications that do not require to hide the size of committed vectors. Non-malleability thus deserves investigation in the case of standard VCs themselves, as pointed out in [58].

Camenisch *et al.* [11, Section 3.1] consider a notion of opening non-malleability in VCs, which requires that multiple local openings of a commitment cannot be combined and maueled into an opening for a different subvector. As such, it prevents aggregating proofs for the same commitment (which is allowed in our setting and in [58]) but does not prevent an adversary from creating commitments and openings to messages related to honest parties’ (which we disallow).

Rotem and Segev [58] showed that Merkle-tree-based VCs [54] are non-malleable in the random oracle model. They also showed that the same result does not hold in the standard model in general. They gave a generic construction of non-malleable VC that combines any VC and a primitive called *locally equivocable commitment with all-but-one binding*, which resembles simulation-sound

trapdoor commitments. Their construction thus inherits its equivocability and local opening properties from separate building blocks. As mentioned earlier, this approach does not extend to the SVC scenario and we thus rely on a single building block (i.e., simulation-sound SVCs) endowed with both properties.

Fleischhacker *et al.* [29] consider simulation-extractability [19, 60] (which is a strong form of non-malleability ensuring knowledge extraction even when the adversary has seen simulated proofs) in aggregatable vector commitments. Their primitive does not imply subvector commitments since it allows aggregating openings for multiple commitments on the same vector positions (instead of multiple positions of the same commitment). Also, it can only be realized in idealized models like the combined algebraic group [30] and random oracle model. In this work, we do not target extractability, which is impossible with succinct proofs in the standard model [36].

## 2 Background and Definitions

NOTATIONS. For positive integers  $a, b$ , the notation  $[a]$  denotes the set  $\{1, \dots, a\}$  while  $[a, b]$  stands for  $\{a, \dots, b\}$ . When  $\mathcal{D}$  is a distribution,  $x \leftarrow \mathcal{D}$  denotes the action of sampling  $x$  according to the distribution  $\mathcal{D}$ . When  $S$  is a finite set, we also denote by  $x \stackrel{\mathcal{R}}{\leftarrow} S$  the action of sampling  $x$  from the uniform distribution over  $S$ . We denote by  $\mathbb{P}_{k,l}$  the set of prime numbers in the interval  $(2^k, 2^l)$ .

When  $\mathbf{m} = (m_1, \dots, m_n) \in \mathcal{M}^n$  is a vector of messages in some space  $\mathcal{M}$  and  $S = \{i_1, \dots, i_{|S|}\} \subseteq [n]$  is a subset of indices,  $\mathbf{m}_S = (m_{i_1}, \dots, m_{i_{|S|}}) \in \mathcal{M}^{|S|}$  denotes the subvector of  $\mathbf{m}$  containing only the positions in  $S$ . We denote by  $\mathbf{m}[S] \in \mathcal{M}^n$  the vector obtained from  $\mathbf{m} \in \mathcal{M}^n$  when replacing all positions in  $[n] \setminus S$  by zeroes. For an index  $i \in [n]$ , we also denote by  $\mathbf{m}[S][i]$  the  $i$ -th coordinate of  $\mathbf{m}[S] \in \mathcal{M}^n$ .

### 2.1 Hardness Assumptions

Our first construction makes use of asymmetric bilinear maps  $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_T$  in groups of prime order  $p$ . We rely on a variant of the computational Bilinear Diffie-Hellman assumption introduced in [9].

**Definition 1.** *In bilinear map groups  $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$  of prime order  $p$ , the **co-squared Bilinear Diffie-Hellman Problem** (co-sqBDH) consists in computing  $e(g, \hat{g})^{ab^2}$  given  $(g, \hat{g}, g^a, g^b, \hat{g}^b)$  for random  $a, b \stackrel{\mathcal{R}}{\leftarrow} \mathbb{Z}_p$ . The co-sqBDH assumption is the intractability of co-sqBDH for any PPT algorithm.*

The hardness of co-sqBDH is implied by that of the standard co-BDH problem.

**Definition 2.** *In bilinear map groups  $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$  of prime order  $p$ , the **co-Bilinear Diffie-Hellman Problem** (co-BDH) consists in computing  $e(g, \hat{g})^{abc}$  given  $(g, \hat{g}, g^a, g^b, g^c, \hat{g}^a, \hat{g}^b, \hat{g}^c)$  for random  $a, b, c \stackrel{\mathcal{R}}{\leftarrow} \mathbb{Z}_p$ .*

Indeed, if we have an oracle that solves co-sqBDH, we can use it to solve a co-BDH instance via three queries on  $(g, \hat{g}, g^{a+b}, \hat{g}^c, \hat{g}^{a+b})$ ,  $(g, \hat{g}, g^a, g^c, \hat{g}^a)$  and  $(g, \hat{g}, g^b, g^c, \hat{g}^b)$ , which yield  $e(g, \hat{g})^{(a+b)^2c}$ ,  $e(g, \hat{g})^{a^2c}$  and  $e(g, \hat{g})^{b^2c}$ , respectively.<sup>6</sup>

In our second construction, we rely on the Strong RSA assumption for an RSA modulus  $N = pq$  that is a product of safe primes  $= 2p' + 1$  and  $q = 2q' + 1$  (i.e., where  $p'$  and  $q'$  are also primes).

**Definition 3** ([2]). *Let a safe-prime product  $N = pq$  where  $p, q$  are primes of at least  $l(\lambda)$  bits for some polynomial  $l : \mathbb{N} \rightarrow \mathbb{N}$ . The Strong RSA assumption states that, for any PPT algorithm  $\mathcal{A}$ , we have*

$$\Pr[y = x^e \bmod N \wedge e > 1 \mid y \stackrel{R}{\leftarrow} \mathbb{Z}_N^*, (x, e) \leftarrow \mathcal{A}(N, y)] = \text{negl}(\lambda)$$

Our third construction relies on the bilinear version of the Strong Diffie-Hellman assumption, where the solution lives in the target group of the pairing.

**Definition 4.** *Let asymmetric bilinear groups  $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$  of prime order  $p$ . For an integer  $q \in \text{poly}(\lambda)$ , the  $q$ -Bilinear Strong Diffie-Hellman ( $q$ -BSDH) problem is, given  $(g, g^\alpha, g^{\alpha^2}, \dots, g^{\alpha^q}), \hat{g}, \hat{g}^\alpha, \hat{g}^{\alpha^2}, \dots, \hat{g}^{\alpha^q})$ , where  $\alpha \stackrel{R}{\leftarrow} \mathbb{Z}_p$ ,  $g \stackrel{R}{\leftarrow} \mathbb{G}$ ,  $\hat{g} \stackrel{R}{\leftarrow} \hat{\mathbb{G}}$ , to find a pair  $(c, e(g, \hat{g})^{1/(\alpha+c)}) \in \mathbb{Z}_p \times \mathbb{G}_T$ .*

## 2.2 Subvector Commitments

We first recall the syntax of vector commitments [14, 46] with the extension of Lai and Malavolta [44], which allows for subvector openings.

**Definition 5.** *A subvector commitment scheme (SVC) (Setup, Commit, Open, Verify) is a tuple of (possibly randomized) algorithms where:*

- **Setup** inputs a security parameter and a vector dimension  $n$ . It outputs a common reference string  $\text{crs}$  that specifies the message space  $\mathcal{M}$  where the vector components live and (optionally) a simulation trapdoor  $\text{tk}$ . The reference string  $\text{crs}$  is implicitly taken as input by all other algorithms hereunder.
- **Commit** is a (possibly randomized) algorithm that inputs a vector  $\mathbf{m} \in \mathcal{M}^n$  and outputs a commitment  $C$  to  $\mathbf{m}$ , together with the state information  $\text{st}$  allowing to open  $C$  later on.
- **Open** is a (possibly randomized) algorithm that inputs a commitment  $C$  together with its corresponding state information  $\text{st}$  and a subset  $S \subseteq [n]$ . It outputs an opening  $\pi_S$ .
- **Verify** is a (usually deterministic) algorithm that inputs a commitment  $C$ , a subset  $S \subseteq [n]$ , a claimed subvector  $\mathbf{m}_S \in \mathcal{M}^{|S|}$ , and a candidate proof  $\pi_S$ . It outputs 0 or 1.

<sup>6</sup> Similar reductions for variants of the Diffie-Hellman problem were given in [1, 53]. In the different calls to the co-sqBDH oracle, the reduction has to randomize the exponents to create independent instances but the idea remains the same.

**Definition 6 (Correctness).** A subvector commitment is correct if, for any security parameter and dimension  $\lambda, n \in \mathbb{N}$ , any  $(\text{crs}, \text{tk}) \leftarrow \text{Setup}(1^\lambda, 1^n)$ , any vector  $\mathbf{m} \in \mathcal{M}^n$ , any subset  $S \subseteq [n]$ , any commitment/state pair obtained as  $(C, \text{st}) \leftarrow \text{Commit}(\text{crs}, \mathbf{m})$ , any opening  $\pi_S \leftarrow \text{Open}(\text{crs}, C, S, \text{st})$ , there is a negligible function  $\epsilon(\lambda)$  such that  $\Pr[\text{Verify}(\text{crs}, S, \mathbf{m}_S, C, \pi_S) = 1] \geq 1 - \epsilon(\lambda)$ .

The compactness property requires that the size of commitments and openings be short, regardless of the dimension of committed vectors and opened subsets.

**Definition 7 (Compactness).** A subvector commitment is compact if there exists a universal  $p(\lambda) \in \text{poly}(\lambda)$  such that, for any  $n \in \text{poly}(\lambda)$ , any  $(\text{crs}, \text{tk}) \leftarrow \text{Setup}(1^\lambda, 1^n)$ , any vector  $\mathbf{m} \in \mathcal{M}^n$ , any subset  $S \subseteq [n]$ , any commitment/state pair obtained as  $(C, \text{st}) \leftarrow \text{Commit}(\text{crs}, \mathbf{m})$ , any opening  $\pi_S \leftarrow \text{Open}(\text{crs}, C, S, \text{st})$ , we have  $|C| \leq p(\lambda)$  and  $|\pi_S| \leq p(\lambda)$ .

**Definition 8 (Position-binding).** A subvector commitment is position-binding if, for any PPT adversary  $\mathcal{A}$ , we have

$$\begin{aligned} \Pr[(\text{crs}, \text{tk}) \leftarrow \text{Setup}(1^\lambda, 1^n); (C, S_1, S_2, \mathbf{m}_{S_1}, \mathbf{m}_{S_2}, \pi_{S_1}, \pi_{S_2}) \leftarrow \mathcal{A}(\text{crs}) : \\ S_1, S_2 \subseteq [n] \wedge \exists i \in S_1 \cap S_2 \text{ s.t. } \mathbf{m}[S_1][i] \neq \mathbf{m}[S_2][i] \\ \wedge \text{Verify}(\text{crs}, S_1, \mathbf{m}_{S_1}, C, \pi_{S_1}) = 1 \\ \wedge \text{Verify}(\text{crs}, S_2, \mathbf{m}_{S_2}, C, \pi_{S_2}) = 1] \leq \text{negl}(\lambda). \end{aligned}$$

So far, the above definitions did not use the trapdoor  $\text{tk}$ . The trapdoor will come into play when we formalize the simulation-sound flavor of subvector commitments, which explicitly requires equivocability.

The definition of non-malleable subvector commitment is a direct adaptation of the definition given by Rotem and Segev [58] in the case of vector commitments. Analogously to [18, 20, 21, 33, 52], this definition considers non-malleability with respect to openings (rather than w.r.t. commitments [23]). This is inevitable in the context of non-interactive VCs, where we cannot properly speak of the “content of a commitment” since commitments are compressing. The main difference with ordinary non-interactive commitments is that the adversary can obtain local openings *before* outputting its own commitment  $\widehat{\text{vcom}}$  at step 6.

**Definition 9.** An SVC is **non-malleable** if, for any  $n \in \text{poly}(\lambda)$ , and any PPT adversary  $\mathcal{A}$ , there is a PPT simulator  $\mathcal{S}$  such that the following holds: For any PPT algorithm  $\mathcal{R}$  and any valid distribution  $\mathcal{D}$  over  $\mathcal{M}^n$ , there exists a negligible function  $\nu(\lambda)$  such that

$$\text{Adv}_{\mathcal{A}, \mathcal{S}, \mathcal{R}, \mathcal{D}}^{\text{nmvc}}(\lambda) \triangleq |\Pr[\text{Real}_{\text{nmVC}, n, \mathcal{A}, \mathcal{D}}(\lambda)] - \Pr[\text{Ideal}_{\text{nmVC}, n, \mathcal{S}, \mathcal{D}}(\lambda)]| \leq \nu(\lambda)$$

where the real and ideal experiments are defined as follows.

**Real**<sub>nmVC,n,A,D</sub>( $\lambda$ ) :

1.  $(\text{crs}, \text{tk}) \leftarrow \text{Setup}(1^\lambda, 1^n)$
2.  $\mathbf{m} = (m_1, \dots, m_n) \leftarrow \mathcal{D}$
3.  $(\text{vcom}, \text{st}) \leftarrow \text{Commit}(\text{crs}, \mathbf{m})$
4.  $(\{S_i\}_{i=1}^t, \text{st}_A) \leftarrow \mathcal{A}(\text{crs}, \text{vcom})$
5. For each  $i \in [t]$ ,  
 $\pi_{S,i} \leftarrow \text{Open}(\text{crs}, \text{vcom}, S_i, \text{st})$
6.  $(\widehat{\text{vcom}}, \{\mathcal{J}_i\}_{i=1}^s, \text{st}_A)$   
 $\leftarrow \mathcal{A}(\text{st}_A, \{\mathbf{m}_{S_i}\}_{i=1}^t, \pi_{S,i})$
7. Let  $S = \cup_{i=1}^t S_i$  and  $\bar{S} = [n] \setminus S$ .
8. For each  $j \in \bar{S}$ ,  
 $\pi_j \leftarrow \text{Open}(\text{crs}, \text{vcom}, \{j\}, \text{st})$
9.  $(\{\widehat{\mathbf{m}}_{\mathcal{J}_i}, \widehat{\pi}_{\mathcal{J}_i}\}_{i=1}^s)$   
 $\leftarrow \mathcal{A}(\text{st}_A, (m_j)_{j \in \bar{S}}, (\pi_j)_{j \in \bar{S}})$
10. If  $\widehat{\text{vcom}} = \text{vcom}$  or  $\exists j \in [s]$  s.t.  
 $\text{Verify}(\text{crs}, \mathcal{J}_i, \widehat{\mathbf{m}}_{\mathcal{J}_i}, \widehat{\text{vcom}}, \widehat{\pi}_{\mathcal{J}_i}) = 0$ ,  
*output*  
 $((m_1, \dots, m_n), (\perp)^{\sum_{i=1}^s |\mathcal{J}_i|},$   
 $(\perp)^{n - |\cup_{i=1}^s \mathcal{J}_i|}, (S_i)_{i=1}^t).$
- Otherwise, *output*  
 $((m_1, \dots, m_n), (\widehat{\mathbf{m}}_{\mathcal{J}_i})_{i=1}^s,$   
 $(\perp)^{n - |\cup_{i=1}^s \mathcal{J}_i|}, (S_i)_{i=1}^t).$

**Ideal**<sub>nmVC,n,S,D</sub>( $\lambda$ ) :

1.  $\mathbf{m} = (m_1, \dots, m_n) \leftarrow \mathcal{D}$
2.  $(\{S_i\}_{i=1}^t, \text{st}_S) \leftarrow \mathcal{S}(1^\lambda, \mathcal{D})$
3.  $(\{\widehat{\mathbf{m}}_{\mathcal{J}_i}, \widehat{\pi}_{\mathcal{J}_i}\}_{i=1}^s)$   
 $\leftarrow \mathcal{S}(\text{st}_S, \{\mathbf{m}[S_i]\}_{i=1}^t)$
4. *Output*  
 $(\mathbf{m}, (\widehat{\mathbf{m}}_{\mathcal{J}_i})_{i=1}^s,$   
 $(\perp)^{n - |\cup_{i=1}^s \mathcal{J}_i|}, (\mathcal{J}_i)_{i=1}^s).$

As in [58], Definition 9 imposes a restriction on the distribution of committed vectors. A distribution  $\mathcal{D}$  over  $\mathcal{M}^n$  is called *valid* if it supports efficient conditional re-sampling: For every  $\mathbf{m} = (m_1, \dots, m_n)$  in the support of  $\mathcal{D}$  and every subset  $S = \{i_1, \dots, i_{|S|}\} \subseteq [n]$ , there is an efficient algorithm that can sample  $\mathbf{m}' \in \mathcal{M}^n$  from the conditional distribution  $\mathcal{D} | (\forall i \in S : \mathbf{m}'[i] = \mathbf{m}[i])$ . As observed in [35, 58], this restriction is necessary since we are facing a selective decommitment problem [24] and we would run into impossibility results [3] without this restriction.

ON OPENINGS FOR POSITIONS OUTSIDE  $\cup_{i=1}^t S_i$ . At step 9 of the real experiment, the adversary is given openings  $\{\pi_j\}_{j \in \bar{S}}$  for all positions  $j \in \bar{S} = [n] \setminus \cup_{i=1}^t S_i$  that have not been opened before  $\mathcal{A}$  declared its subsets  $\{\mathcal{J}_i\}_{i=1}^s$  at step 6. Alternatively, we could allow  $\mathcal{A}$  to choose an arbitrary collection of subsets of  $\bar{S}$  (as part of its output at step 6) and obtain the corresponding openings at step 9. Our schemes remain secure under such a modified definition.

RE-USABILITY. In Definition 9 and the definition of [58], the adversary is only given a single commitment  $\text{vcom}$  at step 4. As pointed out in [18, 33, 52], this is not equivalent to a definition of *re-usable* non-malleable commitments (in the terminology of Damgård and Groth [18]) where the adversary is given multiple honestly generated commitments and tries to open a commitment of its own to messages that are somehow related to honest users' messages.

As mentioned by Rotem and Segev [58] in the context of vector commitments, Definition 9 can be strengthened by giving  $\mathcal{A}$  a set of  $Q \in \text{poly}(\lambda)$  commitments  $\{\text{vcom}_i\}_{i \in [Q]}$  to vectors  $\{\mathbf{m}_i\}_{i \in [Q]}$  independently sampled from  $\mathcal{D}$  at step 4. For

each of these commitments,  $\mathcal{A}$  then has to choose subsets  $\{S_{i,j}\}_{i \in [Q], j \in [t]}$  for which it obtains openings  $(\mathbf{m}_{i,S_{i,j}}, \boldsymbol{\pi}_{i,S_{i,j}})_{i \in [Q], j \in [t]}$  at step 6. At step 6,  $\mathcal{A}$  is then required to output a commitment of its own  $\widetilde{\text{vcom}}$  together with subsets  $\{\mathcal{J}_i\}_{i=1}^s$ . At step 9,  $\mathcal{A}$  obtains openings for all positions  $S_i = [n] \setminus \cup_{j=1}^t S_{i,j}$  that have not been opened for each committed vector  $\mathbf{m}_i$ . Then,  $\mathcal{A}$  is asked to open  $\widetilde{\text{vcom}}$  to subvectors  $\{\widetilde{\mathbf{m}}_{\mathcal{J}_i}\}_{i=1}^s$  that are non-trivially related to  $\{\mathbf{m}_i\}_{i=1}^Q$ . The ideal experiment is modified accordingly by having *Ideal* sample  $Q$  vectors  $\{\mathbf{m}_i\}_{i=1}^Q$  from  $\mathcal{D}$  and allowing  $\mathcal{S}$  to choose  $t$  subsets  $\{S_{i,j}\}_{j \in [t]}$  for each  $i \in [Q]$ . An SVC scheme that satisfies such a definition is called re-usable.

### 2.3 Simulation-Sound Subvector Commitments

As a building block for non-malleable SVCs, we consider an extension of the notion of simulation-sound trapdoor commitments [31, 52] to the context of subvector commitments. As in SSTCs, simulation-sound subvector commitments are vector commitments where each commitment is labeled with a tag. In contrast with standard subvector commitments, they are explicitly required to be hiding since they must be equivocal.

Their main additional security property is called *simulation-sound binding* (or sometimes “simulation-binding” for short). It captures that, even if the adversary can see equivocations of commitments to possibly distinct subsets for several tags  $\text{tag}_1, \dots, \text{tag}_Q$ , it will not be able to break the binding property for a new tag  $\text{tag} \notin \{\text{tag}_1, \dots, \text{tag}_Q\}$ .

**Definition 10** ([52]). *A simulation-sound subvector commitment consists of a tuple of efficient algorithms (ssVC.Setup, ssVC.Commit, ssVC.Open, ssVC.FakeCom, ssVC.FakeOpen, ssVC.Verify) where*

$$(\text{ssVC.Setup}, \text{ssVC.Commit}, \text{ssVC.Open}, \text{SimC.Verify})$$

*forms a subvector commitment scheme and (ssVC.FakeCom, ssVC.FakeOpen) are PPT algorithms with the following properties*

**Trapdoor:** *for any tag  $\in \mathcal{T}$  and any message vector  $\mathbf{m} \in \mathcal{M}^n$  and any subsets  $S_1, \dots, S_t \subseteq [n]$ , the following two distributions are at most  $2^{-\lambda}$  apart in terms of statistical distance:*

$$\begin{aligned} D_{\text{fake}} := & \{(\text{crs}, \text{tk}) \leftarrow \text{ssVC.Setup}(1^\lambda, 1^n); \\ & (\widetilde{\text{vcom}}, \text{aux}) \leftarrow \text{ssVC.FakeCom}(\text{crs}, \text{tk}, \text{tag}); \\ & \forall i \in [t] : \widetilde{\pi}_{S,i} \leftarrow \text{ssVC.FakeOpen}(\text{crs}, \text{tk}, \text{tag}, \widetilde{\text{vcom}}, S_i, \mathbf{m}_{S,i}, \text{aux}) : \\ & (\text{crs}, \text{tag}, \widetilde{\text{vcom}}, \{\mathbf{m}_{S,i}, \widetilde{\pi}_{S,i}\}_{i=1}^t)\} \end{aligned}$$

$$\begin{aligned} D_{\text{real}} := & \{(\text{crs}, \text{tk}) \leftarrow \text{ssVC.Setup}(1^\lambda, 1^n); (\text{vcom}, \text{st}) \leftarrow \text{ssVC.Commit}(\text{crs}, \text{tag}, \mathbf{m}), \\ & \forall i \in [t] : \pi_{S,i} \leftarrow \text{ssVC.Open}(\text{crs}, \text{tag}, \text{vcom}, S_i, \text{st}) : \\ & (\text{crs}, \text{tag}, \text{vcom}, \{\mathbf{m}_{S,i}, \pi_{S,i}\}_{i=1}^t)\}. \end{aligned}$$

**Simulation-sound binding:** for any PPT adversary  $\mathcal{A}$ , the following probability is negligible

$$\Pr[(\text{crs}, \text{tk}) \leftarrow \text{ssVC.Setup}(1^\lambda, 1^n); \\ (\text{vcom}, \text{tag}, S_1, S_2, \mathbf{m}_{S_1}, \mathbf{m}_{S_2}, \pi_{S_1}, \pi_{S_2}) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{tk}, \text{crs}}}(\text{crs}) : \\ \text{tag} \notin \mathcal{Q} \wedge S_1, S_2 \subseteq [n] \wedge \exists i \in S_1 \cap S_2 \text{ s.t. } \mathbf{m}[S_1][i] \neq \mathbf{m}[S_2][i] \\ \wedge \text{ssVC.Verify}(\text{crs}, \text{tag}, S_1, \mathbf{m}_{S_1}, \text{vcom}, \pi_{S_1}) = 1 \\ \wedge \text{ssVC.Verify}(\text{crs}, \text{tag}, S_2, \mathbf{m}_{S_2}, \text{vcom}, \pi_{S_2}) = 1],$$

where  $\mathcal{O}_{\text{tk}, \text{crs}}$  is an oracle that maintains an initially empty set  $\mathcal{Q}$  and operates as follows:

- On input  $(\text{commit}, \text{tag})$ , it runs  $(\widetilde{\text{vcom}}, \text{aux}) \leftarrow \text{ssVC.FakeCom}(\text{crs}, \text{tk}, \text{tag})$ , stores  $(\widetilde{\text{vcom}}, \text{tag}, \text{aux})$ , returns  $\widetilde{\text{vcom}}$  and adds  $\text{tag}$  in a set  $\mathcal{Q}$  (which is initially empty).
- On input  $(\text{open}, \widetilde{\text{vcom}}, \mathbf{m}[S], S)$ : if a tuple  $(\widetilde{\text{vcom}}, \text{tag}, \text{aux})$  was previously stored, it computes  $\tilde{\pi}_S \leftarrow \text{ssVC.FakeOpen}(\text{crs}, \text{tk}, \text{tag}, \widetilde{\text{vcom}}, S, \mathbf{m}_S, \text{aux})$  and returns  $\tilde{\pi}_S$ . Otherwise,  $\mathcal{O}_{\text{tk}, \text{pk}}$  returns  $\perp$ .

Importantly, in the simulation-sound binding experiment, the adversary can make multiple queries  $(\text{open}, \widetilde{\text{vcom}}, \cdot, \cdot)$  for the same fake commitment  $\widetilde{\text{vcom}}$  and arbitrary subsets  $S \subseteq [n]$  and subvectors  $\mathbf{m}_S \in \mathcal{M}^{|S|}$  that may be inconsistent with one another.

**NON-ADAPTIVE-TAG QUERIES.** We also consider a relaxation of the above definition with a *non-adaptive-tag* queries, where the adversary has to declare the set of tags  $\text{tag}_1, \dots, \text{tag}_Q$  for which it wants to query the  $\mathcal{O}_{\text{tk}, \text{crs}}$  oracle before seeing the CRS. Gennaro [33] used such a relaxed notion to achieve non-malleability from similar-looking multi-trapdoor commitments. In our setting, it will be sufficient as well.

**ON UNBOUNDED VS ONE-TIME SIMULATION-BINDING.** Definition 10 is adapted from the definition of simulation-sound binding of [52], which is well-suited to the construction of re-usable non-malleable commitments. The reason is that the adversary is allowed to obtain equivocations on an a priori unbounded number of tags  $\{\text{tag}_i\}_{i=1}^Q$  in the learning phase. A relaxed definition can be obtained by fixing  $Q = 1$  and only allowing  $\mathcal{A}$  to see equivocations for a single tag. This relaxed property, which we call *one-time simulation-binding*, is sufficient to construct non-malleable SVC that are not re-usable.

**ON THE CHOICE OF THE SETS  $S_1, S_2$ .** In the simulation-sound binding experiment, the adversary's final output is allowed to contain distinct sets  $S_1 \neq S_2$  as long as they have a non-empty intersection and  $\mathbf{m}[S_1] \neq \mathbf{m}[S_2]$ . For the purpose of constructing non-malleable SVCs, we could relax the definition and impose  $S_1 = S_2$ , which reflects the notion of *same-set binding* defined in [66]. This allows the possibility of achieving a form of non-malleability in SVCs satisfying the weaker notion of same-set binding. Still, our constructions will satisfy the stronger notion of simulation-sound binding captured by Definition 10.



### 3 Constructions of Simulation-Sound Subvector Commitments

#### 3.1 A Construction from the Co-bilinear Diffie-Hellman Assumption

We describe a simulation-sound version of the subvector commitment proposed by Gorbunov *et al.* [37, Appendix A]

**ssVC.Setup**( $1^\lambda, 1^n$ ): On input of a security parameter  $\lambda$  and a vector dimension  $n \in \text{poly}(\lambda)$ , let  $\ell = O(\lambda)$  and generate the CRS as follows:

1. Choose bilinear groups  $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$  of prime order  $p > 2^{l(\lambda)}$ , for some function  $l : \mathbb{N} \rightarrow \mathbb{N}$ , and  $g \xleftarrow{R} \mathbb{G}$ ,  $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$ .
2. Choose  $u_1, \dots, u_n \xleftarrow{R} \mathbb{Z}_p$  and compute  $U_i = g^{u_i}$  for each  $i \in [n]$ .
3. Choose  $v_1, \dots, v_n \xleftarrow{R} \mathbb{Z}_p$  and compute  $\hat{V}_i = \hat{g}^{v_i}$ ,  $V_i = g^{v_i}$  for each  $i \in [n]$ .
4. For each pair  $(i, j) \in [n] \times [n]$  such that  $i \neq j$ , compute  $W_{i,j} = g^{u_i \cdot v_j}$ .
5. Choose  $\alpha_0, \alpha_1, \dots, \alpha_\ell \xleftarrow{R} \mathbb{Z}_p$  and compute  $H_i = g^{\alpha_i}$ ,  $\hat{H}_i = \hat{g}^{\alpha_i}$  for each  $i \in [0, \ell]$ . Define the hash functions  $H_{\mathbb{G}} : \{0, 1\}^\ell \rightarrow \mathbb{G}$  and  $H_{\hat{\mathbb{G}}} : \{0, 1\}^\ell \rightarrow \hat{\mathbb{G}}$  that map a string  $\text{tag} \in \{0, 1\}^\ell$  to

$$H_{\mathbb{G}}(\text{tag}) = H_0 \cdot \prod_{i=1}^{\ell} H_i^{\text{tag}[i]}, \quad H_{\hat{\mathbb{G}}}(\text{tag}) = \hat{H}_0 \cdot \prod_{i=1}^{\ell} \hat{H}_i^{\text{tag}[i]}$$

which satisfy  $e(H_{\mathbb{G}}(\text{tag}), \hat{g}) = e(g, H_{\hat{\mathbb{G}}}(\text{tag}))$  for any  $\text{tag} \in \{0, 1\}^\ell$ .

6. Define the tag space as  $\mathcal{T} = \{0, 1\}^\ell$ .

Output the common reference string

$$\text{crs} = \left( (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \hat{g}, \{U_i\}_{i \in [n]}, \{V_i\}_{i \in [n]}, \{\hat{V}_i\}_{i \in [n]}, \{H_i\}_{i \in [0, \ell]}, \{\hat{H}_i\}_{i \in [0, \ell]}, \{W_{i,j}\}_{(i,j) \in [n]^2 \setminus \{(i,i)\}_{i=1}^n}, \mathcal{T} \right) \quad (3)$$

and a simulation trapdoor  $\text{tk} = \{(u_i, v_i)\}_{i=1}^n$ .

**ssVC.Commit**( $\text{crs}, \text{tag}, \mathbf{m} = (m_1, \dots, m_n)$ ): In order to commit to a vector  $\mathbf{m} = (m_1, \dots, m_n) \in \mathbb{Z}_p^n$ , parse the CRS as in (3). Choose  $\gamma \xleftarrow{R} \mathbb{Z}_p$  and compute

$$C = g^\gamma \cdot \prod_{i=1}^n U_i^{m_i} \in \mathbb{G}.$$

Output  $\text{vcom} = C \in \mathbb{G}$  together with the state information  $\text{st} = (\gamma, \mathbf{m})$ .

**ssVC.FakeCom**( $\text{crs}, \text{tk}, \text{tag}$ ): Parse the CRS as in (3). Choose a random  $\gamma \xleftarrow{R} \mathbb{Z}_p$  and compute

$$C = g^\gamma \in \mathbb{G}.$$

Output the fake commitment  $\widetilde{\text{com}} = C$  and the state  $\text{aux} = \gamma \in \mathbb{Z}_p$ .

**ssVC.Open**( $\widehat{\text{crs}}, \text{tag}, \text{vcom}, S, \text{st}$ ): given a commitment  $\text{vcom} = C \in \mathbb{G}$ , a tag  $\text{tag} \in \mathcal{T}$ , a state information  $\text{st} = (\gamma, \mathbf{m})$ , and a subset  $S \subseteq [n]$ , choose  $r \xleftarrow{R} \mathbb{Z}_p$  and compute

$$\pi_1 = H_{\mathbb{G}}(\text{tag})^r \cdot \left( \prod_{j \in S} V_j^\gamma \prod_{i \in [n] \setminus \{j\}} W_{i,j}^{m_i} \right), \quad \pi_2 = g^{-r}$$

Return the opening  $\pi_S = (\pi_1, \pi_2) \in \mathbb{G}^2$ .

**ssVC.FakeOpen**( $\widehat{\text{crs}}, \text{tk}, \text{tag}, \widehat{\text{vcom}}, S, \mathbf{m}_S, \text{aux}$ ): given a tag  $\text{tag} \in \mathcal{T}$ , a fake commitment  $\widehat{\text{vcom}} = C$ , a state information  $\text{aux} = \gamma \in \mathbb{Z}_p$ , a subset  $S \subseteq [n]$ , a target subvector  $\mathbf{m}_S = (m_i)_{i \in S} \in \mathcal{M}^{|S|}$ , and an equivocation trapdoor  $\text{tk} = \{(u_i, v_i)\}_{i=1}^n$ , choose  $r \xleftarrow{R} \mathbb{Z}_p$  and compute

$$\pi_1 = H_{\mathbb{G}}(\text{tag})^r \cdot \prod_{j \in S} V_j^\gamma \cdot g^{-\sum_{i \in S} u_i \cdot v_i \cdot m_i}, \quad \pi_2 = g^{-r} \quad (4)$$

Return the simulated opening  $\pi_S = (\pi_1, \pi_2) \in \mathbb{G}^2$ .

**ssVC.Verify**( $\widehat{\text{crs}}, \text{tag}, S, \mathbf{m}_S, \text{vcom}, \pi_S$ ): given a commitment  $\text{vcom} = C \in \mathbb{G}$ , a tag  $\text{tag} \in \{0, 1\}^\ell$ , a subset  $S \subseteq [n]$ , a candidate subvector  $\mathbf{m}_S = (m_j)_{j \in S}$ , and a proof  $\pi_S = (\pi_1, \pi_2) \in \mathbb{G}^2$ , return 1 if the following equality is satisfied and 0 otherwise:

$$e(C, \prod_{j \in S} \hat{V}_j) = e(\pi_1, \hat{g}) \cdot e(\pi_2, H_{\mathbb{G}}(\text{tag})) \cdot \prod_{j \in S} e(U_j, \hat{V}_j)^{m_j} \quad (5)$$

We note that the commitment algorithm does not use the input tag  $\text{tag}$ . As shown in the proof of Theorem 1, this does not affect the simulation-sound binding property.

**CORRECTNESS.** The scheme is correct since we have

$$\begin{aligned} e(C, \prod_{j \in S} \hat{V}_j) &= e(g^\gamma \cdot g^{\sum_{i=1}^n m_i \cdot u_i}, \hat{g}^{\sum_{j \in S} v_j}) \\ &= e\left(\prod_{j \in S} V_j^\gamma, \hat{g}\right) \cdot e\left(g^{\sum_{i=1}^n m_i \cdot u_i}, \hat{g}^{\sum_{j \in S} v_j}\right) \\ &= e\left(\prod_{j \in S} V_j^\gamma, \hat{g}\right) \cdot e\left(\prod_{j \in S} \prod_{i \in [n] \setminus \{j\}}^n g^{m_i \cdot u_i \cdot v_j}, \hat{g}\right) \cdot \prod_{j \in S} e(U_j, \hat{V}_j)^{m_j} \\ &= e\left(\prod_{j \in [S]} V_j^\gamma \prod_{i \in [n] \setminus \{j\}} W_{i,j}^{m_i}, \hat{g}\right) \cdot \prod_{j \in S} e(U_j, \hat{V}_j)^{m_j} \end{aligned} \quad (6)$$

If we now introduce the factors  $H_{\mathbb{G}}(\mathbf{tag})^r$  and  $g^{-r}$  into (6), we obtain the verification equation (5) since

$$\begin{aligned}
 e(C, \prod_{j \in S} \hat{V}_j) &= e\left(\prod_{j \in S} V_i^\gamma \prod_{i \in [n] \setminus \{j\}} W_{i,j}^{m_i}, \hat{g}\right) \cdot \prod_{j \in S} e(U_j, \hat{V}_j)^{m_j} \\
 &= e(H_{\mathbb{G}}(\mathbf{tag})^r \cdot \prod_{j \in S} V_i^\gamma \prod_{i \in [n] \setminus \{j\}} W_{i,j}^{m_i}, \hat{g}) \cdot e(g^{-r}, H_{\hat{\mathbb{G}}}(\mathbf{tag})) \\
 &\quad \cdot \prod_{j \in S} e(U_j, \hat{V}_j)^{m_j} \\
 &= e(\pi_1, \hat{g}) \cdot e(\pi_2, H_{\hat{\mathbb{G}}}(\mathbf{tag})) \cdot \prod_{j \in S} e(U_j, \hat{V}_j)^{m_j}
 \end{aligned}$$

We now prove that the above scheme is unbounded simulation-sound binding: i.e., it remains secure when the adversary obtains equivocations on polynomially many tags before attempting to break the binding property on a different tag. If we just consider the weaker notion of one-time simulation-sound binding, the reduction becomes tighter as the exact security bound (7) is linearly affected by the number of equivocation queries.

**Theorem 1.** *The above construction is a simulation-sound subvector commitment for non-adaptive tag queries under the co-squared Bilinear Diffie-Hellman assumption. For any non-adaptive simulation-sound binding adversary  $\mathcal{A}$  making  $Q$  equivocation queries, there is a PPT co-sqBDH solver  $\mathcal{B}$  such that*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{ssvc}}(\lambda) \leq Q \cdot \ell \cdot n \cdot \mathbf{Adv}_{\mathcal{B}}^{\text{co-sqBDH}}(\lambda) \quad (7)$$

*Proof.* The trapdoor property is straightforward to verify since the distribution of fake commitments and their equivocations is exactly the same as that of real commitments and openings. We thus focus on the simulation-binding property.

The proof is inspired by the simplified security proof given in [41, Theorem 6.1] for Waters signatures [65] when signing queries are non-adaptive.

The reduction  $\mathcal{B}$  is given a co-sqBDH instance  $(g, \hat{g}, g^a, g^b, \hat{g}^b)$  and uses a simulation-sound binding adversary  $\mathcal{A}$  with advantage  $\varepsilon$  to compute  $e(g, \hat{g})^{ab^2}$  with probability  $\varepsilon/(Q \cdot n \cdot \ell)$ .

At the onset of the experiment, the adversary  $\mathcal{A}$  first commits to the tags  $\mathbf{tag}_1, \dots, \mathbf{tag}_Q \in \{0, 1\}^\ell$  for which it plans to ask for equivocations of commitments. Then,  $\mathcal{B}$  chooses  $\mu^* \xleftarrow{R} [Q]$  as a guess that  $\mathbf{tag}_{\mu^*}$  will be the queried tag with the longest common prefix with the target tag  $\mathbf{tag}^*$ . In addition,  $\mathcal{B}$  chooses  $k \xleftarrow{R} [\ell]$  as a guess for the length of the longest common prefix between one of the  $\{\mathbf{tag}_i\}_{i=1}^Q$  and  $\mathbf{tag}^*$ . If  $\mathcal{B}$ 's guesses are correct (which is the case with probability  $1/(Q \cdot \ell)$ ), we will thus have  $\mathbf{tag}_{\mu^*}[1, k] = \mathbf{tag}^*[1, k]$  and  $\mathbf{tag}_{\mu^*}[k+1] \neq \mathbf{tag}^*[k+1]$ . Finally,  $\mathcal{B}$  draws an index  $i^* \xleftarrow{R} [n]$  as a guess that  $i^*$  will be the smallest index such that  $i^* \in S_1 \cap S_2$  and  $\mathbf{m}[S_1][i^*] \neq \mathbf{m}[S_2][i^*]$  in the simulation-sound binding experiment of Definition 10. Having made these guesses,  $\mathcal{B}$  prepares the CRS by

choosing  $\alpha_0, \dots, \alpha_\ell \xleftarrow{R} \mathbb{Z}_p$  and setting

$$\begin{aligned} H_0 &= g^{\alpha_0} \cdot (g^b)^{\sum_{i=1}^k \text{tag}_{\mu^*}[i]}, \\ H_i &= g^{\alpha_i} \cdot (g^b)^{(-1)^{\text{tag}_{\mu^*}[i]}} & i \in [1, k+1] \\ H_i &= g^{\alpha_i} & i \in [k+2, \ell] \\ \hat{H}_0 &= \hat{g}^{\alpha_0} \cdot (\hat{g}^b)^{\sum_{i=1}^k \text{tag}_{\mu^*}[i]} \\ \hat{H}_i &= \hat{g}^{\alpha_i} \cdot (\hat{g}^b)^{(-1)^{\text{tag}_{\mu^*}[i]}} & i \in [1, k+1] \\ \hat{H}_i &= \hat{g}^{\alpha_i} & i \in [k+2, \ell] \end{aligned}$$

Then,  $\mathcal{B}$  generates the rest of the CRS by setting  $U_{i^*} = g^a$ ,  $V_{i^*} = g^b$ ,  $\hat{V}_{i^*} = \hat{g}^b$  and  $U_i = g^{u_i}$ ,  $V_i = g^{v_i}$ ,  $\hat{V}_i = \hat{g}^{v_i}$  with  $u_i, v_i \xleftarrow{R} \mathbb{Z}_p$  for each  $i \in [n] \setminus \{i^*\}$ . This implicitly defines  $u_{i^*} = a$  and  $v_{i^*} = b$ , where  $a, b$  are unknown to  $\mathcal{B}$ . However, from  $(U_{i^*}, V_{i^*})$  and  $\{(u_i, v_i)\}_{i \neq i^*}$ ,  $\mathcal{B}$  can compute  $W_{i,j} = g^{u_i \cdot v_j}$  for each pair  $(i, j) \in [n]^2$  with  $i \neq j$ . It then provides  $\mathcal{A}$  with a CRS containing

$$\left( g, \hat{g}, \{U_i\}_{i \in [n]}, \{V_i\}_{i \in [n]}, \{\hat{V}_i\}_{i \in [n]}, \right. \\ \left. \{H_i\}_{i \in [0, \ell]}, \{\hat{H}_i\}_{i \in [0, \ell]}, \{W_{i,j}\}_{(i,j) \in [n]^2 \setminus \{(i,i)\}_{i=1}^n} \right),$$

which have the correct distribution.

When  $\mathcal{A}$  makes a query (`commit`,  $\text{tag}_i$ ) for one of the tags  $\{\text{tag}_i\}_{i=1}^Q$  that it declared at the very beginning of the game,  $\mathcal{B}$  computes  $C = g^\gamma$  for a random  $\gamma \xleftarrow{R} \mathbb{Z}_p$ , returns  $\widetilde{\text{vcom}} = C$  to  $\mathcal{A}$  and retains  $\text{aux} = \gamma$  for later use. When  $\mathcal{A}$  sends a query (`open`,  $\widetilde{\text{vcom}}$ ,  $\mathbf{m}[S]$ ,  $S$ ),  $\mathcal{B}$  recalls that corresponding tag  $\text{tag}_i$ . If  $i^* \notin S$ ,  $\mathcal{B}$  can run the real `ssVC.FakeOpen` algorithm since it knows  $\{(u_i, v_i)\}_{i \neq i^*}$ . Otherwise (i.e., if  $i^* \in S$ ), it has to simulate `ssVC.FakeOpen` by exploiting the fact that

$$\begin{aligned} H_{\mathbb{G}}(\text{tag}_i) &= H_0 \cdot \prod_{\tau=1}^{\ell} H_i^{\text{tag}_i[\tau]} \\ &= g^{\alpha_0 + \sum_{\tau=1}^{\ell} \alpha_\tau \cdot \text{tag}_i[\tau]} \cdot (g^b)^{\sum_{\tau=1}^k \text{tag}_{\mu^*}[\tau] + (-1)^{\text{tag}_{\mu^*}[\tau]} \cdot \text{tag}_i[\tau]} \\ &= g^{\alpha_0 + \sum_{\tau=1}^{\ell} \alpha_\tau \cdot \text{tag}_i[\tau]} \cdot (g^b)^{\delta[\text{tag}_{\mu^*}, \text{tag}_i]} \end{aligned}$$

where  $\delta[\text{tag}_{\mu^*}, \text{tag}_i] = \sum_{\tau=1}^{k+1} \text{tag}_{\mu^*}[\tau] + (-1)^{\text{tag}_{\mu^*}[\tau]} \cdot \text{tag}_i[\tau] \in \{0, \ell\}$  is the Hamming distance between  $\text{tag}_{\mu^*}[1, k+1] \in \{0, 1\}^{k+1}$  and  $\text{tag}_i[1, k+1] \in \{0, 1\}^{k+1}$ . In the event that  $\delta[\text{tag}_{\mu^*}, \text{tag}_i] = 0$ ,  $\mathcal{B}$  aborts and reports failure. Otherwise, it can compute a pair  $(\sigma_1, \sigma_2) = (g^{ab} \cdot H_{\mathbb{G}}(\text{tag})^{\tilde{r}}, g^{-\tilde{r}})$  using the technique of [6]. To do this, it picks  $r \xleftarrow{R} \mathbb{Z}_p$  and computes

$$\sigma_1 = H_{\mathbb{G}}(\text{tag})^r \cdot (g^a)^{-\frac{\alpha_0 + \sum_{\tau=1}^{\ell} \alpha_\tau \cdot \text{tag}_i[\tau]}{\delta[\text{tag}_{\mu^*}, \text{tag}_i]}}, \quad \sigma_2 = g^{-r} \cdot (g^a)^{\frac{1}{\delta[\text{tag}_{\mu^*}, \text{tag}_i]}}$$

which has the required distribution if we define  $\tilde{r} = r - \frac{a}{\delta[\text{tag}_{\mu^*}, \text{tag}_i]}$ . Armed with the pair  $(\sigma_1, \sigma_2)$ ,  $\mathcal{B}$  can simulate a valid output of `ssVC.FakeOpen` by choosing

$r' \xleftarrow{R} \mathbb{Z}_p$  and computing

$$\begin{aligned}\pi_1 &= \sigma_1^{-m_{i^*}} H_{\mathbb{G}}(\text{tag})^{r'} \cdot \left( \prod_{j \in S \setminus \{i^*\}} V_j^\gamma \right) \cdot g^{-\sum_{i \in S \setminus \{i^*\}} u_i \cdot v_i \cdot m_i} \\ \pi_2 &= \sigma_2^{-m_{i^*}} \cdot g^{-r'}\end{aligned}$$

When  $\mathcal{A}$  halts, it outputs a tuple  $(\text{vcom}, \text{tag}^*, S_1, S_2, \mathbf{m}_{S_1}, \mathbf{m}_{S_2}, \pi_{S_1}, \pi_{S_2})$  comprised of a commitment  $\text{vcom} = C \in \mathbb{G}$ , a tag  $\text{tag}^*$  and possibly distinct sets  $S_1, S_2 \subseteq [n]$  with subvectors  $\mathbf{m}_{S_1} \in \mathbb{Z}_p^{|S_1|}$ ,  $\mathbf{m}_{S_2} \in \mathbb{Z}_p^{|S_2|}$  such that there exists  $i \in S_1 \cap S_2$  satisfying  $\mathbf{m}[S_1][i] \neq \mathbf{m}[S_2][i]$  and  $\pi_{S_1}, \pi_{S_2}$  are valid subvector openings. At this point,  $\mathcal{B}$  fails if  $\text{tag}^*[1, k+1] \neq \text{tag}_{\mu^*}[1, k+1]$  (which happens if it incorrectly guessed the longest common prefix between  $\text{tag}^*$  and  $\{\text{tag}_i\}_{i=1}^Q$ ) or  $\mathbf{m}[S_1][i^*] \neq \mathbf{m}[S_2][i^*]$  (which means that it incorrectly guessed the smallest index  $i^*$  that would satisfy  $\mathcal{A}$ 's winning conditions). If  $\mathcal{B}$  does not fail, we have  $H_{\mathbb{G}}(\text{tag}^*) = g^{\alpha_0 + \sum_{\tau=1}^{\ell} \alpha_\tau \cdot \text{tag}^*[\tau]}$  and  $\mathcal{B}$  managed to embed its co-sqBDH input in the correct  $U_{i^*} = g^a$ ,  $V_{i^*} = g^b$  such that  $i^* \in S_1 \cap S_2$ .

Since  $\pi_{S_1} = (\pi_1, \pi_2) \in \mathbb{G}^2$  and  $\pi_{S_2} = (\pi'_1, \pi'_2) \in \mathbb{G}^2$  are valid openings, we have

$$\begin{aligned}e(C, \prod_{j \in S_1} \hat{V}_j) &= e(\pi_1, \hat{g}) \cdot e(\pi_2, H_{\hat{\mathbb{G}}}(\text{tag}^*)) \cdot \prod_{j \in S_1} e(U_j, \hat{V}_j)^{\mathbf{m}[S_1][j]} \\ e(C, \prod_{j \in S_2} \hat{V}_j) &= e(\pi'_1, \hat{g}) \cdot e(\pi'_2, H_{\hat{\mathbb{G}}}(\text{tag}^*)) \cdot \prod_{j \in S_2} e(U_j, \hat{V}_j)^{\mathbf{m}[S_2][j]}.\end{aligned}$$

Since  $H_{\hat{\mathbb{G}}}(\text{tag}^*) = \hat{g}^{\alpha_0 + \sum_{\tau=1}^{\ell} \alpha_\tau \cdot \text{tag}^*[\tau]}$ , this can be written

$$e(C, \prod_{j \in S_1} \hat{V}_j) = e(\pi, \hat{g}) \cdot \prod_{j \in S_1} e(U_j, \hat{V}_j)^{\mathbf{m}[S_1][j]} \quad (8)$$

$$e(C, \prod_{j \in S_2} \hat{V}_j) = e(\pi', \hat{g}) \cdot \prod_{j \in S_2} e(U_j, \hat{V}_j)^{\mathbf{m}[S_2][j]}. \quad (9)$$

where  $\pi = \pi_1 \cdot \pi_2^{\alpha_0 + \sum_{\tau=1}^{\ell} \alpha_\tau \cdot \text{tag}^*[\tau]}$  and  $\pi' = \pi'_1 \cdot \pi'_2^{\alpha_0 + \sum_{\tau=1}^{\ell} \alpha_\tau \cdot \text{tag}^*[\tau]}$ . If we raise (8) to the power  $\sum_{i \in S_2} v_i$  and (9) to the power  $\sum_{i \in S_1} v_i$ , we get

$$\begin{aligned}e(\pi^{\sum_{i \in S_2} v_i}, \hat{g}) \cdot \prod_{j \in S_1} e(U_j, \hat{V}_j)^{\mathbf{m}[S_1][j] \cdot (\sum_{i \in S_2} v_i)} \\ = e(\pi'^{\sum_{i \in S_1} v_i}, \hat{g}) \cdot \prod_{j \in S_2} e(U_j, \hat{V}_j)^{\mathbf{m}[S_2][j] \cdot (\sum_{i \in S_1} v_i)}\end{aligned}$$

If we now isolate the term  $i^* \in S_1 \cap S_2$  from other terms, the previous equality becomes

$$\begin{aligned}e(\pi^{\sum_{i \in S_2} v_i}, \hat{g}) \cdot e(U_{i^*}, \hat{V}_{i^*})^{\mathbf{m}[S_1][i^*] \cdot (\sum_{i \in S_2} v_i)} \cdot \prod_{j \in S_1 \setminus \{i^*\}} e(U_j, \hat{V}_j)^{\mathbf{m}[S_1][j] \cdot (\sum_{i \in S_2} v_i)} \\ = e(\pi'^{\sum_{i \in S_1} v_i}, \hat{g}) \cdot e(U_{i^*}, \hat{V}_{i^*})^{\mathbf{m}[S_2][i^*] \cdot (\sum_{i \in S_1} v_i)} \cdot \prod_{j \in S_2 \setminus \{i^*\}} e(U_j, \hat{V}_j)^{\mathbf{m}[S_2][j] \cdot (\sum_{i \in S_1} v_i)}\end{aligned}$$

or, equivalently,

$$\begin{aligned}
& e(\pi^{\sum_{i \in S_2} v_i}, \hat{g}) \cdot e(U_{i^*}, \hat{V}_{i^*})^{\mathbf{m}[S_1][i^*] \cdot (\sum_{i \in S_2 \setminus \{i^*\}} v_i)} \cdot e(U_{i^*}, \hat{V}_{i^*})^{\mathbf{m}[S_1][i^*] \cdot v_{i^*}} \\
& \quad \cdot \prod_{j \in S_1 \setminus \{i^*\}} e(U_j, \hat{V}_j)^{\mathbf{m}[S_1][j] \cdot (\sum_{i \in S_2} v_i)} \\
& = e(\pi^{\sum_{i \in S_1} v_i}, \hat{g}) \cdot e(U_{i^*}, \hat{V}_{i^*})^{\mathbf{m}[S_2][i^*] \cdot (\sum_{i \in S_1 \setminus \{i^*\}} v_i)} \cdot e(U_{i^*}, \hat{V}_{i^*})^{\mathbf{m}[S_2][i^*] \cdot v_{i^*}} \\
& \quad \cdot \prod_{j \in S_2 \setminus \{i^*\}} e(U_j, \hat{V}_j)^{\mathbf{m}[S_2][j] \cdot (\sum_{i \in S_1} v_i)}
\end{aligned}$$

Since  $\mathbf{m}[S_1][i^*] \neq \mathbf{m}[S_2][i^*]$  by hypothesis, the above equality allows  $\mathcal{B}$  to compute  $e(U_{i^*}, \hat{V}_{i^*})^{v_{i^*}} = e(g, \hat{g})^{ab^2}$  as

$$\begin{aligned}
e(g, \hat{g})^{ab^2} & = \left( e(\pi^{\sum_{i \in S_2} v_i} / \pi^{\sum_{i \in S_1} v_i}, \hat{g}) \right. \\
& \quad \cdot e(U_{i^*}, \hat{V}_{i^*})^{\mathbf{m}[S_1][i^*] \cdot (\sum_{i \in S_2 \setminus \{i^*\}} v_i) - \mathbf{m}[S_2][i^*] \cdot (\sum_{i \in S_1 \setminus \{i^*\}} v_i)} \\
& \quad \cdot \prod_{j \in S_1 \setminus \{i^*\}} e(U_j, \hat{V}_j)^{\mathbf{m}[S_1][j] \cdot (\sum_{i \in S_2} v_i)} \\
& \quad \left. \cdot \prod_{j \in S_2 \setminus \{i^*\}} e(U_j, \hat{V}_j)^{-\mathbf{m}[S_2][j] \cdot (\sum_{i \in S_1} v_i)} \right)^{1/(\mathbf{m}[S_1][i^*] - \mathbf{m}[S_2][i^*])},
\end{aligned}$$

where the factors of the right-hand-side member are all computable since

$$\begin{aligned}
e(\pi^{\sum_{i \in S_2} v_i}, \hat{g}) & = e(\pi^{\sum_{i \in S_2 \setminus \{i^*\}} v_i}, \hat{g}) \cdot e(\pi, \hat{V}_{i^*}) \\
e(\pi^{\sum_{i \in S_1} v_i}, \hat{g}) & = e(\pi^{\sum_{i \in S_1 \setminus \{i^*\}} v_i}, \hat{g}) \cdot e(\pi', \hat{V}_{i^*}),
\end{aligned}$$

$$\begin{aligned}
\prod_{j \in S_1 \setminus \{i^*\}} e(U_j, \hat{V}_j)^{\mathbf{m}[S_1][j] \cdot (\sum_{i \in S_2} v_i)} & = e(V_{i^*}, \hat{g})^{(\sum_{j \in S_1 \setminus \{i^*\}} \mathbf{m}[S_1][j] \cdot u_j \cdot v_j)} \\
& \quad \cdot \prod_{j \in S_1 \setminus \{i^*\}} e(U_j, \hat{V}_j)^{\mathbf{m}[S_1][j] \cdot (\sum_{i \in S_2 \setminus \{i^*\}} v_i)},
\end{aligned}$$

and

$$\begin{aligned}
\prod_{j \in S_2 \setminus \{i^*\}} e(U_j, \hat{V}_j)^{\mathbf{m}[S_2][j] \cdot (\sum_{i \in S_1} v_i)} & = e(V_{i^*}, \hat{g})^{(\sum_{j \in S_2 \setminus \{i^*\}} \mathbf{m}[S_2][j] \cdot u_j \cdot v_j)} \\
& \quad \cdot \prod_{j \in S_1 \setminus \{i^*\}} e(U_j, \hat{V}_j)^{\mathbf{m}[S_2][j] \cdot (\sum_{i \in S_1 \setminus \{i^*\}} v_i)}
\end{aligned}$$

To conclude the proof, we note that  $\mathcal{B}$  does not fail as long as it correctly guesses  $\mu^* \in [Q]$ ,  $k \in [\ell]$  and  $i^* \in [n]$ . Since these indexes are chosen uniformly and independently of  $\mathcal{A}$ 's view, they are correct with probability  $1/(Q \cdot \ell \cdot n)$ , which yields the stated upper bound on  $\mathcal{A}$ 's advantage.  $\square$

**SAME-COMMITMENT AGGREGATION.** We note that the scheme retains the same-commitment aggregation property of the scheme described by Gorbunov *et al.* [37, Appendix A]. Namely, on input of openings of a given commitment  $C$  for disjoint subsets  $S_1, S_2 \subseteq [n]$ , it is possible to publicly compute a merged proof for the union  $S_1 \cup S_2$  and the same commitment  $C$ .

### 3.2 A Construction from the Strong RSA Assumption

We describe a construction from the Strong RSA assumption. The scheme is based on the Lai-Malavolta subvector commitment [44], which builds itself on the RSA-based VC of Catalano and Fiore [14]. In order to achieve simulation-soundness, we adapt a technique used by Gennaro [33].

**ssVC.Setup**( $1^\lambda, 1^n$ ): On input of a security parameter  $\lambda$  and a vector dimension  $n \in \text{poly}(\lambda)$ , generate the CRS as follows:

1. Choose a safe prime product  $N = pq$  where  $p = 2p' + 1$ ,  $q = 2q' + 1$  for primes  $p, q, p'q' > 2^{l(\lambda)}$  for some polynomial  $l \in \text{poly}(\lambda)$ .
2. Choose a random quadratic residue  $g_0 \stackrel{R}{\leftarrow} \mathbb{QR}_N$ .
3. Define the message space as  $\mathcal{M} = \{0, 1\}^k$  and the tag space as  $\mathcal{T} = \{0, 1\}^\ell$ , for some integers  $k, \ell \in \text{poly}(\lambda)$  such that  $k < l$ .
4. Choose a hash function  $H : \mathcal{T} \times [n] \rightarrow \mathbb{P}_{k,l}$  that maps inputs to prime numbers in the interval  $(2^k, 2^l)$ , where  $k \in \text{poly}(\lambda)$ .

Output the common reference string

$$\text{crs} = \left( N, g_0, H, \mathcal{M}, \mathcal{T} \right) \quad (10)$$

and a simulation trapdoor  $\text{tk} = (p, q)$ .

**ssVC.Commit**( $\text{crs}, \text{tag}, \mathbf{m} = (m_1, \dots, m_n)$ ): In order to commit to a vector  $\mathbf{m} = (m_1, \dots, m_n) \in \mathcal{M}^n$ , parse the CRS as in (10) and do the following.

1. For each  $i \in [n]$ , compute  $e_i = H(\text{tag}, i) \in \mathbb{P}_{k,l}$  to obtain a set of primes  $\{e_i\}_{i=1}^n$  such that  $e_i \in (2^k, 2^l)$  for each  $i \in [n]$ .
2. Compute  $g = g_0^{\prod_{i=1}^n e_i} \bmod N$ . For each  $i \in [n]$ , compute

$$g_i = g_0^{\prod_{j \in [n] \setminus \{i\}} e_j} \bmod N.$$

3. Choose a random  $\gamma \stackrel{R}{\leftarrow} \mathbb{Z}_{(N-1)/4}$  and compute

$$C = g^\gamma \cdot \prod_{i=1}^n g_i^{m_i} \bmod N.$$

Output the commitment  $\text{vcom} = C \in \mathbb{Z}_N^*$  together with the state information  $\text{st} = (\gamma, \mathbf{m}, (e_i)_{i \in [n]})$ .

**ssVC.FakeCom**( $\text{crs}, \text{tk}, \text{tag}$ ): Parse the CRS as in (10) and do the following.

1. For each  $i \in [n]$ , compute a prime  $e_i = H(\text{tag}, i) \in \mathbb{P}_{k,l}$ . Then, compute  $g = g_0^{\prod_{i=1}^n e_i} \bmod N$ .

2. Choose a random  $\gamma \xleftarrow{R} \mathbb{Z}_{(N-1)/4}$  and compute

$$C = g^\gamma \bmod N.$$

Output the fake commitment  $\widetilde{\text{vcom}} = C \in \mathbb{Z}_N^*$  as well as the state information  $\text{aux} = (\gamma, (e_i)_{i \in [n]})$ .

**ssVC.Open**( $\text{crs}, \text{tag}, \text{vcom}, S, \text{st}$ ): given a commitment  $\text{vcom} = C \in \mathbb{Z}_N^*$ , a tag  $\text{tag} \in \mathcal{T}$ , a state information  $\text{st} = (\gamma, \mathbf{m}, (e_i)_{i \in [n]})$ , and a subset  $S \subseteq [n]$ ,

1. Define  $e_S = \prod_{i \in S} e_i$ .
2. Compute  $\pi_S = (C \cdot \prod_{i \in S} g_i^{-m_i})^{1/e_S} \bmod N$  as

$$\pi_S = g_0^{\gamma \cdot \prod_{i \in [n] \setminus S} e_i} \cdot \prod_{i \in [n] \setminus S} g_0^{m_i \cdot (\prod_{j \in [n] \setminus (S \cup \{i\})} e_j)} \bmod N \quad (11)$$

Return the opening  $\pi_S \in \mathbb{Z}_N^*$ .

**ssVC.FakeOpen**( $\text{crs}, \text{tk}, \text{tag}, \widetilde{\text{vcom}}, S, \mathbf{m}_S, \text{aux}$ ): given a tag  $\text{tag} \in \mathcal{T}$ , a fake commitment  $\widetilde{\text{vcom}} = C \in \mathbb{Z}_N^*$  with its state information  $\text{aux} = (\gamma, \{e_i\}_{i \in [n]})$ , a subset  $S \subseteq [n]$ , a target subvector  $\mathbf{m}_S = (m_i)_{i \in S} \in \mathcal{M}^{|S|}$ , and an equivocation trapdoor  $\text{tk} = (p, q)$ ,

1. Define  $e_S = \prod_{i \in S} e_i$ . For each  $i \in S$ , compute  $g_i = g_0^{\prod_{j \in [n] \setminus \{i\}} e_j} \bmod N$ .
2. Use  $\text{tk} = (p, q)$  to compute and return

$$\pi_S = (C \cdot \prod_{i \in S} g_i^{-m_i})^{1/e_S} \bmod N \quad (12)$$

**ssVC.Verify**( $\text{crs}, \text{tag}, S, \mathbf{m}_S, \text{vcom}, \pi_S$ ): given a commitment  $\text{vcom} = C \in \mathbb{Z}_N^*$ , a tag  $\text{tag} \in \{0, 1\}^\ell$ , a subset  $S \subseteq [n]$ , a candidate subvector  $\mathbf{m}_S = (m_j)_{j \in S}$ , and a candidate proof  $\pi_S \in \mathbb{Z}_N^*$ ,

1. For each  $i \in [n]$ , compute  $e_i = H(\text{tag}, i) \in \mathbb{P}_{k,l}$ . Define  $e_S = \prod_{i \in S} e_i$ .
2. For each  $i \in S$ , compute  $g_i = g_0^{\prod_{j \in [n] \setminus \{i\}} e_j} \bmod N$ .
3. Return 1 if and only if the following equality is satisfied:

$$C = \pi_S^{e_S} \cdot \prod_{i \in S} g_i^{m_i} \bmod N \quad (13)$$

**CORRECTNESS.** We note that the equivocation algorithm **ssVC.FakeOpen** can always compute an  $e_S$ -th root in (12) since  $\gcd(e_S, \varphi(N)) = 1$  (due the requirement that  $2^k < e_i < 2^l < \min(p', q')$  for each  $i \in [n]$ ).

The correctness of **ssVC.Open** follows from the fact that honestly generated commitments satisfy

$$\begin{aligned} C \cdot \prod_{i \in S} g_i^{-m_i} &= g^\gamma \cdot \prod_{i \in [n] \setminus S} g_i^{m_i} \bmod N \\ &= g_0^{\gamma \cdot \prod_{i \in [n]} e_i} \cdot \prod_{i \in [n] \setminus S} g_0^{m_i \cdot \prod_{j \in [n] \setminus \{i\}} e_j} \bmod N, \end{aligned}$$



which immediately shows that (11) is the valid opening.

We now prove that the scheme is unbounded simulation-binding under the Strong RSA assumption and assuming the collision resistance of  $H$ .

The proof relies on Lemma 1, which is sometimes attributed to [61], but follows from a standard application of the extended Euclidean algorithm.

**Lemma 1** ([61]). *Let  $G$  be a group. Suppose that  $e_1, e_2 \in \mathbb{Z}$  are co-prime integers. Given  $a, b \in G$  such that  $a^{e_1} = b^{e_2}$ , one can compute  $g$  such that  $g^{e_2} = a$  using  $O(\log(e_1 + e_2))$  group and arithmetic operations.*

**Theorem 2.** *The above construction is a simulation-sound subvector commitment for non-adaptive tag queries under the Strong RSA assumption and assuming that  $H$  is collision-resistant. For any non-adaptive simulation-sound binding adversary  $\mathcal{A}$ , there exists either a PPT algorithm  $\mathcal{B}_0$  that computes collisions on  $H$  or a PPT algorithm  $\mathcal{B}_1$  that solves the Strong RSA problem such that*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{ssvc}}(\lambda) \leq \mathbf{Adv}_{\mathcal{B}_0}^{\text{Coll}}(\lambda) + \mathbf{Adv}_{\mathcal{B}_1}^{\text{SRSA}}(\lambda)$$

*Proof.* We first consider the trapdoor property and observe that fake commitments (just like real commitments) are statistically uniform in the cyclic subgroup  $\mathbb{QR}_N$  of quadratic residues. Moreover, for any real or fake commitment  $C$ , any subset  $S \subseteq [n]$  and any subvector  $\mathbf{m}_S$ , there is only one proof  $\pi_S$  (determined by (17)) satisfying the verification equation (13) since  $\gcd(e_S, \varphi(N)) = 1$ . We now turn to the simulation-sound binding property and prove it assuming the collision-resistance of  $H$  and under the strong RSA assumption.

Algorithm  $\mathcal{B}$  is given (keys for) a hash function  $H$  and a strong RSA instance  $(N, y)$ , with a random  $y \in \mathbb{Z}_N^*$ . It uses a simulation-sound binding adversary  $\mathcal{A}$  with non-negligible advantage  $\varepsilon$  to compute a non-trivial root of  $y$  with advantage  $\varepsilon/2$  or a collision on  $H$  with advantage  $\varepsilon/2$ .

At the outset of the experiment, the adversary  $\mathcal{A}$  first announces the tags  $\text{tag}_1, \dots, \text{tag}_Q \in \{0, 1\}^\ell$  on which it will make equivocation queries. For each  $\text{tag}_i \in \mathcal{T}$ ,  $\mathcal{B}$  computes  $e_{\text{tag}_i, \tau} = H(\text{tag}_i, \tau)$  for all  $\tau \in [n]$  and uses these sets primes  $\{(e_{\text{tag}_i, \tau})_{\tau \in [n]}\}_{i=1}^Q$  to define

$$g_0 = y^{2U} \bmod N$$

where  $U = \prod_{i=1}^Q \prod_{\tau=1}^n e_{\text{tag}_i, \tau} \in \mathbb{Z}$ . It then includes  $(N, g_0, H)$  in  $\text{crs}$ , which is returned to  $\mathcal{A}$  at the beginning of the game.

Whenever  $\mathcal{A}$  makes a query ( $\text{commit}, \text{tag}_i$ ) for one of the tags  $\{\text{tag}_i\}_{i=1}^Q$  chosen upfront,  $\mathcal{B}$  runs the real  $\text{ssVC.FakeCom}$  algorithm. Namely, it defines  $g = g_0^{\prod_{\tau=1}^n e_{\text{tag}_i, \tau}}$ , computes  $C = g^\gamma \bmod N$  for a random  $\gamma \xleftarrow{R} \mathbb{Z}_{(N-1)/4}$ , returns  $\widetilde{\text{vcom}} = C$  to  $\mathcal{A}$  and retains  $\text{aux} = \gamma$  for later use. When  $\mathcal{A}$  sends a query ( $\text{open}, \widetilde{\text{vcom}}, \mathbf{m}[S], S$ ),  $\mathcal{B}$  recalls that corresponding tag  $\text{tag}_i$  and the primes  $(e_{\text{tag}_i, \tau})_{\tau \in [n]}$ . Let  $e_{S,i} = \prod_{\tau \in S} e_{\text{tag}_i, \tau}$ . We remark that  $\mathcal{B}$  can efficiently compute

$$\pi_S = (C \cdot \prod_{i \in S} g_i^{-m_i})^{1/e_{S,i}} = \left( g_0^{\gamma \cdot \prod_{\tau=1}^n e_{\text{tag}_i, \tau}} \cdot \prod_{i \in S} g_0^{-m_i \cdot \prod_{\tau \in [n] \setminus \{i\}} e_{\text{tag}_i, \tau}} \right)^{1/e_{S,i}} \bmod N$$

since it knows

$$g_0^{1/e_{S,i}} = y^{2(\prod_{\tau \in [n] \setminus S} e_{\text{tag}_i, \tau}) \cdot (\prod_{j \in [Q] \setminus \{i\}} \prod_{\tau=1}^n e_{\text{tag}_j, \tau})} \pmod{N}.$$

Hence, due to the way  $g_0$  was set up in the preparation phase,  $\mathcal{B}$  is always able to consistently answer equivocation queries.

When  $\mathcal{A}$  terminates, its output  $(\text{vcom}, \text{tag}^*, S_1, S_2, \mathbf{m}_{S_1}, \mathbf{m}_{S_2}, \pi_{S_1}, \pi_{S_2})$  is expected to contain a commitment  $\text{vcom} = C \in \mathbb{Z}_N^*$ , a fresh tag  $\text{tag}^*$  and possibly distinct sets  $S_1, S_2 \subseteq [n]$  with subvectors  $\mathbf{m}_{S_1} \in \mathcal{M}^{|S_1|}$ ,  $\mathbf{m}_{S_2} \in \mathcal{M}^{|S_2|}$  for which  $\pi_{S_1}, \pi_{S_2}$  are valid subvector openings although there exists  $i \in S_1 \cap S_2$  satisfying  $\mathbf{m}[S_1][i] \neq \mathbf{m}[S_2][i]$ .

At this point,  $\mathcal{B}$  computes  $e_{\text{tag}^*, \tau} = H(\text{tag}^*, \tau)$  for each  $\tau \in [n]$ . If there exists  $(i, j) \in [Q] \times [n]$  such that  $H(\text{tag}_i, j) = H(\text{tag}^*, \tau)$  for some  $\tau \in [n]$ , then  $\mathcal{B}$  has found a collision on  $H$  since  $\mathcal{A}$ 's winning conditions impose that  $\text{tag}^* \notin \{\text{tag}_i\}_{i=1}^Q$ . Otherwise, we have  $\gcd(\prod_{\tau=1}^n e_{\text{tag}^*, \tau}, U) = 1$ .

Since  $\pi_{S_1}, \pi_{S_2}$  are valid proofs, we must have

$$C = \pi_{S_1}^{e_{S_1}} \cdot \prod_{i \in S_1} g_i^{\mathbf{m}[S_1][i]} \pmod{N} = \pi_{S_2}^{e_{S_2}} \cdot \prod_{i \in S_2} g_i^{\mathbf{m}[S_2][i]} \pmod{N} \quad (14)$$

where  $e_{S_1} = \prod_{i \in S_1} e_{\text{tag}^*, i}$ ,  $e_{S_2} = \prod_{i \in S_2} e_{\text{tag}^*, i}$  and  $g_i = g_0^{\prod_{j \in [n] \setminus \{i\}} e_{\text{tag}^*, j}} \pmod{N}$  for each  $i \in [n]$ . Let an arbitrary  $i^* \in S_1 \cap S_2$  such that  $\mathbf{m}[S_1][i^*] \neq \mathbf{m}[S_2][i^*]$ . We assume w.l.o.g. that  $\mathbf{m}[S_1][i^*] < \mathbf{m}[S_2][i^*]$  when they are interpreted as integers, so that  $0 < \mathbf{m}[S_2][i^*] - \mathbf{m}[S_1][i^*] < 2^k$ . Then, (14) implies

$$\begin{aligned} & g_{i^*}^{\mathbf{m}[S_2][i^*] - \mathbf{m}[S_1][i^*]} \\ &= \pi_{S_1}^{e_{S_1}} \cdot \prod_{i \in S_1 \setminus \{i^*\}} g_i^{\mathbf{m}[S_1][i]} \cdot \pi_{S_2}^{-e_{S_2}} \cdot \prod_{i \in S_2 \setminus \{i^*\}} g_i^{-\mathbf{m}[S_2][i]} \pmod{N} \\ &= \left( \pi_{S_1}^{\prod_{i \in S_1 \setminus \{i^*\}} e_{\text{tag}^*, i}} \cdot \prod_{i \in S_1 \setminus \{i^*\}} g_0^{\mathbf{m}[S_1][i] \cdot \prod_{j \in [n] \setminus \{i, i^*\}} e_{\text{tag}^*, j}} \right. \\ & \quad \left. \cdot \pi_{S_2}^{-\prod_{i \in S_2 \setminus \{i^*\}} e_{\text{tag}^*, i}} \cdot \prod_{i \in S_2 \setminus \{i^*\}} g_0^{-\mathbf{m}[S_2][i] \cdot \prod_{j \in [n] \setminus \{i, i^*\}} e_{\text{tag}^*, j}} \right)^{e_{\text{tag}^*, i^*}} \quad (15) \end{aligned}$$

We then observe that

$$\begin{aligned} \Lambda_{\text{tag}^*} \triangleq & \pi_{S_1}^{\prod_{i \in S_1 \setminus \{i^*\}} e_{\text{tag}^*, i}} \cdot \prod_{i \in S_1 \setminus \{i^*\}} g_0^{\mathbf{m}[S_1][i] \cdot \prod_{j \in [n] \setminus \{i, i^*\}} e_{\text{tag}^*, j}} \\ & \cdot \pi_{S_2}^{-\prod_{i \in S_2 \setminus \{i^*\}} e_{\text{tag}^*, i}} \cdot \prod_{i \in S_2 \setminus \{i^*\}} g_0^{-\mathbf{m}[S_2][i] \cdot \prod_{j \in [n] \setminus \{i, i^*\}} e_{\text{tag}^*, j}}, \end{aligned}$$

is computable by  $\mathcal{B}$  from  $\mathcal{A}$ 's output and satisfies

$$g_0^{(\mathbf{m}[S_2][i^*] - \mathbf{m}[S_1][i^*]) \cdot \prod_{j \in [n] \setminus \{i^*\}} e_{\text{tag}^*, j}} = \Lambda_{\text{tag}^*}^{e_{\text{tag}^*, i^*}} \pmod{N}.$$

Since  $\gcd(e_{\text{tag}^*, i^*}, (\mathbf{m}[S_2][i^*] - \mathbf{m}[S_1][i^*]) \cdot \prod_{j \in [n] \setminus \{i^*\}} e_{\text{tag}^*, j}) = 1$ ,  $\mathcal{B}$  can apply Lemma 1 to compute  $\Omega_{\text{tag}^*}$  such that

$$g_0 = \Omega_{\text{tag}^*}^{e_{\text{tag}^*, i^*}} \pmod{N}$$

Next, recalling the way  $g_0$  was defined, we see that the obtained  $\Omega_{\text{tag}^*}$  satisfies

$$y^{2U} = \Omega_{\text{tag}^*}^{e_{\text{tag}^*, i^*}} \pmod{N}$$

where  $U = \prod_{i=1}^Q \prod_{\tau=1}^n e_{\text{tag}_i, \tau}$  is such that  $\gcd(\prod_{\tau=1}^n e_{\text{tag}^*, \tau}, 2U) = 1$  (and thus  $\gcd(e_{\text{tag}^*, i^*}, 2U) = 1$ ). By applying Lemma 1 one more time,  $\mathcal{B}$  eventually obtains  $z_{\text{tag}^*} \in \mathbb{Z}_N^*$  such that

$$y = z_{\text{tag}^*}^{e_{\text{tag}^*, i^*}} \pmod{N},$$

which yields a valid solution  $(e_{\text{tag}^*, i^*}, z_{\text{tag}^*})$  to the strong RSA instance.  $\square$

REMARK. The requirement of  $N$  being a safe-prime product can be relaxed modulo slight changes. For the functionality of `ssVC.FakeOpen` and the proof of Theorem 2, we need to make sure that: (i) All outputs of  $H$  are co-prime with  $\varphi(N)$ ; (ii)  $\mathbb{Z}_N^*$  contains a sufficiently large cyclic subgroup of odd order, where  $g_0$  should be efficiently samplable without knowing the factors of  $N$ .

EFFICIENCY. As in [14], the scheme has a constant-size CRS. On the other hand, its commitment algorithm requires  $O(n \cdot \log n)$  exponentiations with exponents of size  $\max_{i \in [n]} |e_i|$  if the recursive multi-exponentiation algorithm of [8, Section 3.3] is used. The opening algorithm can be optimized in the same way as mentioned in [8, Section 5.3].

In the full version of the paper, we show a more efficient construction satisfying the weaker notion of one-time simulation-binding under the RSA assumption. In this variant, the CRS size is  $O(n)$  and the commitment phase has linear complexity in  $n$  as well.

INCREMENTAL AGGREGATION. The scheme retains the incremental aggregation property of [12] (albeit without constant verification time). Given openings  $\pi_I, \pi_J$  of a commitment to subsets  $I, J \in [n]$ , the technique of [12, Section 5.2] still allows publicly computing an opening  $\pi_{I \cup J}$  to the union  $I \cup J$ .

### 3.3 A Construction from the Bilinear Strong Diffie-Hellman Assumption

We provide a description of the (Bilinear) Strong Diffie-Hellman [7] analogue of our Strong-RSA-based commitment.

The resulting scheme has an  $O(n)$ -size CRS, but the commitment and opening algorithms require  $O(n \cdot \log^3 n)$  field multiplications. However, the number of exponentiations to compute a commitment or an opening is only linear in  $n$ . Also, the online complexity of the commitment algorithm is  $O(n)$ .

The main advantage over the scheme of Sect. 3.1 is to provide a linear-size CRS instead of a quadratic-size one (as well as shorter openings comprised of only one element of  $\mathbb{G}$ ). On the other hand, it relies on a stronger  $q$ -type assumption, where the parameter  $q$  of Definition 4 is  $q = n \cdot (Q + 1)$  if the adversary is allowed to make equivocation queries for  $Q$  distinct tags.

**ssVC.Setup**( $1^\lambda, 1^n$ ): On input of a security parameter  $\lambda$  and a vector dimension  $n \in \text{poly}(\lambda)$ , generate the CRS as follows:

1. Choose pairing-friendly groups  $(\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T)$  of prime order  $p > 2^{l(\lambda)}$ , for some polynomial  $l \in \text{poly}(\lambda)$ .
2. Choose generators  $g \xleftarrow{R} \mathbb{G}$ ,  $\hat{g} \xleftarrow{R} \hat{\mathbb{G}}$ . Then, choose a random  $\alpha \xleftarrow{R} \mathbb{Z}_p$  and compute  $g^{(\alpha^i)}$ ,  $\hat{g}^{(\alpha^i)}$  for each  $i \in [n]$ .
3. Define the message space as  $\mathcal{M} = \mathbb{Z}_p$  and the tag space as  $\mathcal{T} = \mathbb{Z}_p$ .
4. Choose a collision-resistant hash function  $H : \mathcal{T} \times [n] \rightarrow \mathbb{Z}_p^*$ .

Output the common reference string

$$\text{crs} = \left( (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T), g, \hat{g}, (g^{(\alpha^i)})_{i \in [n]}, (\hat{g}^{(\alpha^i)})_{i \in [n]}, H, \mathcal{M}, \mathcal{T} \right) \quad (16)$$

and a simulation trapdoor  $\text{tk} = \alpha$ .

**ssVC.Commit**( $\text{crs}, \text{tag}, \mathbf{m} = (m_1, \dots, m_n)$ ): Given  $\mathbf{m} = (m_1, \dots, m_n) \in \mathbb{Z}_p^n$ , parse the CRS as in (16) and do the following.

1. Compute the polynomial  $p[X] = \prod_{i \in [n]} (X + H(\text{tag}, i))$
2. For each  $i \in [n]$ , compute the quotient polynomial

$$p_i[X] = p[X] / (X + H(\text{tag}, i)) = \prod_{j \in [n] \setminus \{i\}} (X + H(\text{tag}, j)).$$

3. Choose  $\gamma \xleftarrow{R} \mathbb{Z}_p$  and compute

$$C = g^{\gamma \cdot p(\alpha) + \sum_{i=1}^n m_i \cdot p_i(\alpha)}$$

using  $(g, (g^{(\alpha^i)})_{i \in [n]})$ .

Output the commitment  $\text{vcom} = C \in \mathbb{G}$  together with the state information  $\text{st} = (\gamma, \mathbf{m})$ .<sup>7</sup>

**ssVC.FakeCom**( $\text{crs}, \text{tk}, \text{tag}$ ): Parse the CRS as in (16) and do the following.

1. Compute the polynomial  $p[X] = \prod_{i \in [n]} (X + H(\text{tag}, i))$
2. Choose a random  $\gamma \xleftarrow{R} \mathbb{Z}_p$  and compute  $C = g^{\gamma \cdot p(\alpha)}$ .

Output the fake commitment  $\widetilde{\text{vcom}} = C \in \mathbb{G}$  as well as  $\text{aux} = \gamma$ .

**ssVC.Open**( $\text{crs}, \text{tag}, \text{vcom}, S, \text{st}$ ): given a commitment  $\text{vcom} = C \in \mathbb{G}$ , a tag  $\text{tag} \in \mathcal{T}$ , a state information  $\text{st} = (\gamma, \mathbf{m})$ , and a subset  $S \subseteq [n]$ ,

1. Compute  $p_S[X] = \prod_{i \in S} (X + H(\text{tag}, i))$  and  $p_{0,S}[X] = p[X] / p_S[X]$ .

<sup>7</sup> We may also store the coefficients of  $\gamma \cdot p[X] + \sum_{i=1}^n m_i \cdot p_i[X]$  in the state  $\text{st}$ .

2. For each  $i \in [n]$ , compute

$$p_{i,S}[X] = \prod_{j \in [n] \setminus (S \cup \{i\})} (X + H(\mathbf{tag}, j)) = p_{0,S}[X] / (X + H(\mathbf{tag}, i)).$$

3. Compute  $\pi_S = g^{\gamma \cdot p_{0,S}(\alpha) + \sum_{i \in [n] \setminus S} m_i \cdot p_{i,S}(\alpha)}$ .

Return the proof  $\pi_S \in \mathbb{G}$ .

**ssVC.FakeOpen**( $\text{crs}, \text{tk}, \text{tag}, \widetilde{\text{vcom}}, S, \mathbf{m}_S, \text{aux}$ ): given a tag  $\text{tag} \in \mathcal{T}$ , a fake commitment  $\widetilde{\text{vcom}} = C \in \mathbb{G}$  with its state information  $\text{aux} = \gamma$ , a subset  $S \subseteq [n]$ , a target subvector  $\mathbf{m}_S = (m_i)_{i \in S} \in \mathbb{Z}_p^{|S|}$ , and  $\text{tk} = \alpha$ ,

1. Define  $p_S[X] = \prod_{i \in S} (X + H(\mathbf{tag}, i))$ . For each  $i \in [S]$ , compute

$$p_i[X] = \prod_{j \in [n] \setminus \{i\}} (X + H(\mathbf{tag}, j)).$$

2. Compute

$$\pi_S = \left( C \cdot \prod_{i \in S} g^{-m_i \cdot p_i(\alpha)} \right)^{1/p_S(\alpha)} \quad (17)$$

Return the proof  $\pi_S \in \mathbb{G}$ .

**ssVC.Verify**( $\text{crs}, \text{tag}, S, \mathbf{m}_S, \text{vcom}, \pi_S$ ): given a commitment  $\text{vcom} = C \in \mathbb{G}$ , a tag  $\text{tag} \in \mathcal{T}$ , a subset  $S \subseteq [n]$ , a candidate subvector  $\mathbf{m}_S = (m_i)_{i \in S}$ , and a candidate proof  $\pi_S \in \mathbb{G}$ ,

1. For each  $i \in [n]$ , compute  $p_i[X] = \prod_{j \in [n] \setminus \{i\}} (X + H(\mathbf{tag}, j))$ . Compute  $p_S[X] = \prod_{i \in S} (X + H(\mathbf{tag}, i))$ .
2. For each  $i \in S$ , compute  $g_i = g^{\prod_{j \in [n] \setminus \{i\}} (\alpha + H(\mathbf{tag}, j))}$ .
3. Return 0 if the following equality is not satisfied:

$$e\left(C \cdot \prod_{i \in S} g_i^{-m_i}, \hat{g}\right) = e(\pi_S, \hat{g}^{p_S(\alpha)}) \quad (18)$$

Otherwise, return 1.

Correctness follows from a simple inspection of the scheme.

**Theorem 3.** *The above construction is a simulation-sound subvector commitment for non-adaptive tag queries under the Bilinear Strong Diffie-Hellman assumption and assuming that  $H$  is collision-resistant. For any non-adaptive simulation-binding adversary  $\mathcal{A}$  making up to  $Q$  equivocation queries, there exists either a PPT algorithm  $\mathcal{B}_0$  that computes collisions on  $H$  or a PPT algorithm  $\mathcal{B}_1$  that solves the  $n(Q+1)$ -BSDH problem such that*

$$\text{Adv}_{\mathcal{A}}^{\text{ssvc}}(\lambda) \leq \text{Adv}_{\mathcal{B}_0}^{\text{Coll}}(\lambda) + \text{Adv}_{\mathcal{B}_1}^{(n \cdot (Q+1))\text{-BSDH}}(\lambda)$$

(The proof is given in the full version of the paper.)

EFFICIENCY. In the Commit algorithm, each of the polynomials  $(p[X], \{p_i[X]\}_{i=1}^n)$  can be computed in time  $O(n \cdot \log^2 n)$  (as computing the coefficients of a polynomial from its roots can be done in  $O(n \cdot \log^2 n)$  time using the FFT).

Naively computing the linear combination  $\gamma \cdot p[X] + \sum_{i=1}^n m_i \cdot p_i[X]$  would take  $O(n^2)$  multiplications. Fortunately, it is possible to adapt the divide-and-conquer approach of the MultiExp algorithm of Boneh *et al.* [8, Section 3.3] so as to compute  $\gamma \cdot p[X] + \sum_{i=1}^n m_i \cdot p_i[X]$  using  $O(n \cdot \log^3 n)$  multiplications<sup>8</sup> in  $\mathbb{Z}_p$ . This optimization also applies to the Open algorithm.

SPEEDING-UP VERIFICATION. In the random oracle model, it is possible to reduce the verifier's workload to  $O(n \cdot \log^3 n)$  field operations, and  $O(1)$  exponentiations or pairings (and thus eliminate the  $O(n)$  exponentiations). To do this, we can outsource the computation of  $\prod_{i \in S} g_i^{-m_i}$  and  $\hat{g}^{p_S(\alpha)}$  to the prover.

The prover can convince the verifier that these exponentiations were correctly computed by interpreting them as KZG commitments [42] to polynomials  $-\sum_{i \in S} -m_i \cdot p_i[X]$  and  $p_S[X]$ , which it evaluates on a random input derived from a random oracle. Since the verifier knows the polynomials  $-\sum_{i \in S} -m_i \cdot p_i[X]$  and  $p_S[X]$ , it can evaluate them itself and check the prover's KZG evaluation proofs for these evaluations. In the security proof, we can rely on the result of [51], which shows that KZG commitments are knowledge-sound in the random oracle model (under a  $q$ -type assumption) when evaluated on random inputs.

## 4 Non-malleable Subvector Commitments from Simulation-Sound Ones

We now describe a construction of non-malleable subvector commitment. It resembles the generic construction of non-malleable VC from [58] with the difference that it does not use an all-but-one binding locally equivocal commitment. Instead, it uses a tag-based simulation-sound SVC. It relies on the following building blocks.

- A strongly unforgeable one-time signature  $\Sigma = (\text{Sig.Gen}, \text{Sig.Sign}, \text{Sig.Verify})$  with verification key space  $\mathcal{VK}$ .
- A family  $\mathcal{H} = \{h : \mathcal{VK} \rightarrow \{0, 1\}^\ell\}$  of target collision-resistant hash functions [56] with output length  $\ell \in \text{poly}(\lambda)$ .
- A simulation-sound subvector commitment with tag space  $\mathcal{T} = \{0, 1\}^\ell$  for some  $\ell \in \text{poly}(\lambda)$ .

**nmVC.Setup** $(1^\lambda, 1^n)$ : On input of a security parameter  $\lambda$  and a vector dimension  $n \in \text{poly}(\lambda)$ , let  $\ell = O(\lambda)$  and generate the CRS as follows:

1. Run  $(\text{crs}_{\text{ssvc}}, \text{tk}_{\text{ssvc}}) \leftarrow \text{ssVC.Setup}(1^\lambda, 1^n)$  to generate a CRS and a simulation trapdoor for the simulation-sound subvector commitment with tag space  $\mathcal{T} = \{0, 1\}^\ell$ , with  $\ell = \text{poly}(\lambda)$ .

<sup>8</sup> Each of the  $\log n$  recursive steps computes two multi-products of at most  $n/2$  linear polynomials using  $O(n \cdot \log^2 n)$  multiplications, followed by two multiplications with polynomials obtained from earlier recursive steps.

2. Choose the specification of a one-time signature scheme  $\Sigma = (\text{Sig.Gen}, \text{Sig.Sign}, \text{Sig.Verify})$  with key space  $\mathcal{VK}$ .
3. Choose a target collision-resistant hash function  $h \xleftarrow{R} \mathcal{H}$  with domain  $\mathcal{VK}$  and range  $\mathcal{T} = \{0, 1\}^\ell$ .

Output the common reference string  $\text{crs} = (\text{crs}_{\text{ssvc}}, \Sigma, h)$  and a simulation trapdoor  $\text{tk}_{\text{ssvc}}$ .

**nmVC.Commit**( $\text{crs}, \mathbf{m} = (m_1, \dots, m_n)$ ): To commit to  $\mathbf{m} = (m_1, \dots, m_n) \in \mathcal{M}^n$ , parse the CRS as above and do the following.

1. Generate a one-time signature key pair  $(\text{vk}, \text{sk}) \leftarrow \text{Sig.Gen}(1^\lambda)$  and compute a tag  $\tau = h(\text{vk}) \in \mathcal{T}$ .
2. Compute  $(\text{vcom}_{\text{ssvc}}, \text{st}_{\text{ssvc}}) \leftarrow \text{ssVC.Commit}(\text{crs}_{\text{ssvc}}, \tau, \mathbf{m})$ .

Then, output the commitment  $\text{vcom} = (\text{vk}, \text{vcom}_{\text{ssvc}})$  and the state information  $\text{st} = (\text{st}_{\text{ssvc}}, \text{sk})$ .

**nmVC.Open**( $1^\lambda, \text{crs}, \text{vcom}, S, \text{st}$ ): given  $\text{vcom} = (\text{vk}, \text{vcom}_{\text{ssvc}})$ , a state information  $\text{st} = (\text{st}_{\text{ssvc}}, \text{sk})$ , and a subset  $S \subseteq [n]$ , do the following:

1. Compute  $\pi_S \leftarrow \text{ssVC.Open}(\text{crs}, \tau, \text{vcom}_{\text{ssvc}}, S, \text{st})$ , where  $\tau = h(\text{vk}) \in \mathcal{T}$ .
2. Compute  $\sigma_S \leftarrow \text{Sig.Sign}(\text{sk}, \text{vcom}_{\text{ssvc}})$

Return the proof  $\pi_S = (\pi_S, \sigma_S)$ .

**nmVC.Verify**( $\text{crs}, S, \mathbf{m}_S = (m_i)_{i \in S}, \text{vcom}, \pi_S$ ): given  $\text{vcom} = (\text{vk}, \text{vcom}_{\text{ssvc}})$ , a subset  $S \subseteq [n]$ , a candidate subvector  $\mathbf{m}_S = (m_i)_{i \in S}$ , and a proof  $\pi_S$ ,

1. Compute  $\tau = h(\text{vk}) \in \mathcal{T}$  and return 0 if

$$\text{ssVC.Verify}(\text{crs}, \tau, S, \mathbf{m}_S, \text{vcom}_{\text{ssvc}}, \pi_S) = 0.$$

2. Return 1 if  $\text{Sig.Verify}(\text{vk}, \text{vcom}_{\text{ssvc}}) = 1$  and 0 otherwise.

The proof of Theorem 4 is given in the full version of the paper. It follows closely the proof of Rotem and Segev [58], but it extends to provide re-usability if the underlying SSVC is unbounded simulation-sound binding (instead of just one-time simulation-sound binding).

**Theorem 4.** *The above subvector commitment construction is non-malleable assuming that: (i) The underlying SSVC scheme is one-time simulation-sound binding; (ii)  $\Sigma$  is a strongly unforgeable one-time signature; (iii)  $\mathcal{H}$  is a target collision-resistant hash family. For any PPT adversary  $\mathcal{A}$  and any  $n \in \text{poly}(\lambda)$ , there exists a PPT simulator  $\mathcal{S}_A$  such that: For any PPT distinguisher  $\mathcal{R}$ , there are PPT algorithms  $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$  such that*

$$\text{Adv}_{\mathcal{A}, \mathcal{S}_A, \mathcal{R}, \mathcal{D}}^{\text{nmvc}}(\lambda) \leq \text{Adv}_{\mathcal{B}_1}^{\text{forge}}(\lambda) + \text{Adv}_{\mathcal{B}_2}^{\text{Coll}}(\lambda) + 2 \cdot \text{Adv}_{\mathcal{B}_3}^{\text{ssvc}}(\lambda) + 2^{-\lambda}$$

Moreover, if the SSVC scheme is unbounded simulation-binding, the resulting non-malleable subvector commitment is re-usable.

**Acknowledgements.** The author thanks the anonymous reviewers for useful comments.

## References

1. F. Bao, R. Deng, and H. Zhu. Variations of Diffie-Hellman problem. In *ICICS*, 2003.
2. N. Baric and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Eurocrypt*, 1997.
3. M. Bellare, D. Hofheinz, and S. Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *Eurocrypt*, 2009.
4. E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *IEEE S&P*, 2014.
5. M. Bichler. *Market design: A Linear Programming Approach to Auctions and Matching*. 2017.
6. D. Boneh and X. Boyen. Efficient selective identity-based encryption without random oracles. In *Eurocrypt*, 2004.
7. D. Boneh and X. Boyen. Short signatures without random oracles. In *Eurocrypt*, 2004.
8. D. Boneh, B. Büinz, and B. Fisch. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In *Crypto*, 2019.
9. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Crypto*, 2001.
10. H. Brenner, V. Goyal, S. Richelson, A. Rosen, and M. Vald. Fast non-malleable commitment. In *CCS*, 2015.
11. J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss. Composable and modular anonymous credentials: Definitions and practical constructions. In *Asiacrypt*, 2015.
12. M. Campanelli, D. Fiore, N. Greco, D. Kolonelos, and L. Nizzardo. Incrementally aggregatable vector commitments and applications to verifiable decentralized storage. In *Asiacrypt*, 2020.
13. R. Canetti and M. Fischlin. Universally composable commitments. In *Crypto*, 2001.
14. D. Catalano and D. Fiore. Vector commitments and their applications. In *PKC*, 2013.
15. D. Catalano, D. Fiore, R. Gennaro, and E. Giunta. On the impossibility of algebraic vector commitments in pairing-free groups. In *TCC*, 2022.
16. D. Catalano, D. Fiore, and M. Messina. Zero-knowledge sets with short proofs. In *Eurocrypt*, 2008.
17. M. Chase, A. Healy, M. Lysyanskaya, T. Malkin, and L. Reyzin. Mercurial commitments with applications to zero-knowledge sets. In *Eurocrypt*, 2005.
18. I. Damgård and J. Groth. Non-interactive and reusable non-malleable commitment schemes. In *STOC*, 2003.
19. A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. Robust non-interactive zero-knowledge. In *Crypto*, 2001.
20. G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Non-interactive and non-malleable commitments. In *STOC*, 1998.
21. G. Di Crescenzo, J. Katz, R. Ostrovsky, and A. Smith. Efficient and non-interactive non-malleable commitment. In *Eurocrypt*, 2001.
22. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *STOC*, 1991.
23. D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. *SIAM J. of Computing*, 30(2), 2000.



24. C. Dwork, M. Naor, O. Reingold, and L. Stockmeyer. Magic functions. *J. of the ACM*, 50(6), 2003.
25. B. Fisch. PoReps: Proofs of Space on Useful Data. Cryptology ePrint Archive Report 2018/678.
26. M. Fischlin. Trapdoor commitment schemes and their applications. PhD thesis, University of Frankfurt, 2001.
27. M. Fischlin and R. Fischlin. Efficient non-malleable commitment schemes. In *Crypto*, 2000.
28. M. Fischlin and R. Fischlin. The representation problem based on factoring. In *CT-RSA*, 2002.
29. N. Fleischhacker, M. Hall-Andersen, M. Simkin, and B. Wagner. Jackpot: Non-interactive aggregatable lotteries. In *Asiacrypt*, 2024.
30. G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *Crypto*, 2018.
31. J. Garay, P. MacKenzie, and K. Yang. Strengthening zero-knowledge protocols using signatures. In *Eurocrypt*, 2003.
32. R. Garg, D. Khurana, G. Lu, and B. Waters. Black-box non-interactive non-malleable commitments. In *Eurocrypt*, 2021.
33. R. Gennaro. Multi-trapdoor commitments and their applications to non-malleable protocols. In *Crypto*, 2004.
34. R. Gennaro, S. Halevi, and T. Rabin. Secure hash-and-sign signatures without the random oracle. In *Eurocrypt*, 1999.
35. R. Gennaro and S. Micali. Independent zero-knowledge sets. In *ICALP*, 2006.
36. C. Gentry and D. Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, 2011.
37. S. Gorbunov, L. Reyzin, H. Wee, and Z. Zhang. PointProofs: Aggregating Proofs for Multiple Vector Commitments. In *ACM-CCS*, 2020.
38. G. Goyal, C.-K. Lee, R. Ostrovsky, and I. Visconti. Constructing non-malleable commitments: A black-box approach. In *FOCS*, 2012.
39. V. Goyal, O. Pandey, and S. Richelson. Textbook non-malleable commitments. In *STOC*, 2016.
40. V. Goyal, S. Richelson, A. Rosen, and M. Vald. An algebraic approach to non-malleability. In *FOCS*, 2014.
41. S. Hohenberger and B. Waters. Short and stateless signatures from the RSA assumption. In *Crypto*, 2009.
42. A. Kate, G. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and applications. In *Asiacrypt*, 2010.
43. J. Krupp, D. Schröder, M. Simkin, D. Fiore, G. Ateniese, and S. Nuernberger. newblock Nearly optimal verifiable data streaming. In *PKC*, 2016.
44. R.-W. Lai and G. Malavolta. Subvector commitments with application to succinct arguments. In *Crypto*, 2019.
45. D. Leung, Y. Gilad, S. Gorbunov, L. Reyzin, and N. Zeldovich. Aardvark: A concurrent authenticated dictionary with short proof. In *USENIX Security*, 2022.
46. B. Libert and M. Yung. Concise mercurial vector commitments and independent zero-knowledge sets with short proofs. In *TCC*, 2010.
47. H. Lin and R. Pass. Non-malleability amplification. In *FOCS*, 2009.
48. H. Lin and R. Pass. Constant-round non-malleable commitments from any one-way function. In *STOC*, 2011.
49. H. Lin, R. Pass, and P. Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In *FOCS*, 2017.

50. H. Lin, R. Pass, and M. Venkatasubramanian. Concurrent non-malleable commitments from any one-way function. In *TCC*, 2008.
51. H. Lipmaa, R. Parisella, and J. Siim. Constant-size zk-SNARKs in ROM from falsifiable assumptions. In *Eurocrypt*, 2024.
52. P. MacKenzie and K. Yang. On simulation-sound trapdoor commitments. In *Eurocrypt*, 2004.
53. U. Maurer and S. Wolf. Diffie-Hellman oracles. In *Crypto*, 1996.
54. R. Merkle. A certified digital signature. In *Crypto*, 1989.
55. S. Micali, M. Rabin, and J. Kilian. Zero-knowledge sets. In *FOCS*, 2003.
56. M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, 1989.
57. R. Pass and H. Wee. Constant-round non-malleable commitments from sub-exponential one-way functions. In *Eurocrypt*, 2010.
58. L. Rotem and G. Segev. Non-malleable vector commitments via local equivocability. In *TCC*, 2021.
59. L. Rotem and G. Segev. Non-malleable vector commitments via local equivocability. *Jo. of Cryptology*, 36(4), 2023.
60. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, 1999.
61. A. Shamir. On the generation of cryptographically strong pseudorandomsequences. *ACM Transactions on Computer Systems*, 1(1):38–44, 1983.
62. S. Srinivasan, A. Chepurnoy, C. Papamanthou, A. Tomescu, and Y. Zhang. Hyper-proofs: Aggregating and maintaining proofs in vector commitments. In *USENIX Security*, 2022.
63. A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, and D. Khovratovich. Aggregatable subvector commitments for stateless cryptocurrencies. In *SCN*, 2020.
64. A. Tomescu, Y. Xia, and Z. Newman. Authenticated dictionaries with cross-incremental proof (dis)aggregation. Cryptology ePrint Archive Report 2020/1239.
65. B. Waters. Efficient identity-based encryption without random oracles. In *Eurocrypt*, 2005.
66. H. Wee and D. Wu. Succinct vector, polynomial, and functional commitments from lattices. In *Eurocrypt*, 2023.



# Traitor Tracing Without Trusted Authority from Registered Functional Encryption

Pedro Branco<sup>1(✉)</sup>, Russell W. F. Lai<sup>2</sup>, Monosij Maitra<sup>3</sup>, Giulio Malavolta<sup>1</sup>, Ahmadreza Rahimi<sup>4</sup>, and Ivy K. Y. Woo<sup>2</sup>

<sup>1</sup> Bocconi University, Milan, Italy

pedrodemelobranco@gmail.com

<sup>2</sup> Aalto University, Espoo, Finland

<sup>3</sup> Indian Institute of Technology Kharagpur, Kharagpur, India

<sup>4</sup> Max Planck Institute for Security and Privacy, Bochum, Germany

**Abstract.** Traitor-tracing systems allow identifying the users who contributed to building a rogue decoder in a broadcast environment. In a traditional traitor-tracing system, a key authority is responsible for generating the global public parameters and issuing secret keys to users. All security is lost if the *key authority itself* is corrupt. This raises the question: Can we construct a traitor-tracing scheme, without a trusted authority?

In this work, we propose a new model for traitor-tracing systems where, instead of having a key authority, users could generate and register their own public keys. The public parameters are computed by aggregating all user public keys. Crucially, the aggregation process is *public*, thus eliminating the need of any trusted authority. We present two new traitor-tracing systems in this model based on bilinear pairings. Our first scheme is proven adaptively secure in the generic group model. This scheme features a *transparent* setup, ciphertexts consisting of  $6\sqrt{L} + 4$  group elements, and a public tracing algorithm. Our second scheme supports a bounded collusion of traitors and is proven selectively secure in the standard model. Our main technical ingredients are new registered functional encryption (RFE) schemes for quadratic and linear functions which, prior to this work, were known only from indistinguishability obfuscation.

To substantiate the practicality of our approach, we evaluate the performance a proof of concept implementation. For a group of  $L = 1024$  users, encryption and decryption take roughly 50 ms and 4 ms, respectively, whereas a ciphertext is of size 6.7 KB.

## 1 Introduction

Traitor-tracing systems [10] allow identifying the users who contributed to building a rogue decoder in a broadcast environment. In a traditional traitor-tracing system, a key authority is responsible for generating the global public parameters and issuing user secret keys. Given the public parameters, it is possible to encrypt a message so that any user in possession of a secret key can decrypt it. As in standard broadcast encryption, the encrypted message is hidden from any unauthorized user, i.e. those who do not have access to any secret key.

The most important property of a traitor-tracing system, however, is the presence of a *tracing algorithm* which identifies corrupt users. More specifically, if an attacker produces a device that can decrypt ciphertexts with some non-negligible probability, then the tracing algorithm, given black-box access to the device, is guaranteed to identify at least one corrupt user, i.e. a member of those who contributed to the creation of the decryption device.

Traditional traitor-tracing systems [3, 7–9, 23–25, 30, 36, 41] focus on the settings where an arbitrary set of users can be corrupt, assuming that the key authority is honest. Notably, all guarantees are lost if *the key authority itself* is corrupt. This limits the use cases of traditional traitor-tracing systems in the sense that the key authority must either be the same party as the encryptor or trusted by the latter. Even in the latter case, this limitation is clearly undesirable from a security perspective, as it introduces a single point of failure. In fact, we speculate that this limitation has played a significant role in preventing the adoption of traitor-tracing systems in practice, as it is identical in spirit to the key-escrow problem of identity-based encryption [37].

In light of the above limitation of traditional traitor-tracing systems, a natural question is whether we can remove the trust assumption on the key authority.

Can we construct *efficient* traitor-tracing without a trusted authority?

Specifically, we are interested in constructions that achieve non-trivial efficiency (i.e. ciphertext size sublinear in the number of users  $L$ ) and are based on simple and well-understood cryptographic structure, such as bilinear groups.

*A New Model For Traitor Tracing.* We envision a new model for traitor tracing without any trusted authority, that we refer to as *registered traitor-tracing*. In our model, each user samples its own pair of public and secret keys locally (relative to a *preferably unstructured* common reference string), without needing any interaction with any other users. Upon collecting all the public keys  $(pk_1, \dots, pk_L)$ , for instance in a public directory or bulletin-board, it is possible to *aggregate* them into a *short* master public key  $mpk$ . Given  $mpk$ , anyone can encrypt a message  $m$  in such a way that only the registered users are able to recover it. Crucially, the aggregation of the public keys is a completely *transparent and deterministic* process, and therefore *no* trusted party is needed to perform this operation.<sup>1</sup> This model is directly inspired by recent works on registration-based encryption [12, 15–17, 19, 20, 22, 27, 43] and distributed broadcast [9, 18, 31, 40] that adopt a similar paradigm to solve the key escrow problem in related settings.

The distinguishing property of traitor tracing system is (*public*) traceability: If a malicious user  $i^*$  builds a decryption box  $D$ , it should be possible (for *anyone*) to track  $i^*$  given only black-box access to  $D$ . To substantiate the usefulness of this primitive, we discuss how it can be useful in some recurrent scenarios below.

---

<sup>1</sup> Anyone can re-evaluate the aggregation process and check if the output is identical to what is claimed.

*Application: Traceable Group Messaging.* In a group messaging system, a group of  $L$  users wants to broadcast messages to each other privately. Given that messages are constantly exchanged, it is important that the size of the ciphertext should be sublinear in  $L$ , especially for large groups. As the simplest notion of security, we want that an external observer learns no information about the messages exchanged within the group. Furthermore, we want to protect against users *leaking* their secret key, for instance by having their device compromised: To this end, one needs to be able to trace the users corresponding to the leaked key, in order to exclude them from the group.<sup>2</sup>

Superficially, it may appear that group messaging is the killer application for traitor-tracing systems. However, at present, no existing system uses traitor-tracing techniques to build their protocols. We speculate that this is due to the presence of a trusted authority: The cost of adding the tracing property to the system is to insert a trusted authority that can potentially decrypt all messages of all groups! On the other hand, in *registered traitor-tracing* no such tradeoff is present, and we can add traceability to groups without introducing any backdoor. We envision that register-traitor tracing can be used as a cryptographic building block in messaging schemes to add a traceability guarantee, which is not present in current systems. At the time of writing, the maximum size of a Whatsapp group is  $L = 1024$ , which is well within the range of practicality of our scheme.

We remark that secure messaging systems are complex ecosystems that involve substantial research effort to build. Tracing traitors (particularly without a trusted setup) in any such system adds to its complexity. Therefore, we do not claim that integrating (registered) traitor-tracing into such a system would solve all of its related challenges. Rather, it would further need a holistic analysis of the overall system with different trade-offs. However, we view building registered traitor-tracing as an important milestone opening the possibility of using it in real-world secure messaging.

## 1.1 Our Contributions

We construct traitor-tracing systems without a trusted authority, where users can sample their own keys locally without interaction. Formally, we introduce a new primitive called *registered traitor-tracing* (RTT) supporting an unbounded collusion of traitors. We then present two constructions in the bounded and unbounded collusion settings.

Our main technical ingredients are new constructions of *registered functional encryption* (RFE) for quadratic and linear functions from bilinear groups. Prior works either built general-purpose RFE based on indistinguishability obfuscation (iO) [12, 17], or specialised RFE for *inner-product predicates* from bilinear groups [17, 42].<sup>3</sup> In more detail, our contributions are summarised as follows:

<sup>2</sup> Our definition applies to cases where the key is publicly leaked, or sold over the Internet. In which case, one could buy the key and easily implement the decoder box by just running the decryption algorithm.

<sup>3</sup> We note that there have been some concurrent works in this direction which we will discuss in Sect. 1.2.

(1) *A New Model for Traitor-Tracing.* We introduce the notion of registered traitor-tracing (RTT) as a new model to build traitor-tracing systems without a trusted authority. We propose appropriate security definitions for RTT and show compilers (inspired by the literature in non-registered setting) that allows us to reduce the problem of constructing RTT to building a weak form of RFE for quadratic functions and an RFE for linear functions. We also discuss efficient strategies to revoke traitors.<sup>4</sup>

(2) *Unbounded-Collusion RTT.* We propose a new RFE for *quadratic* functions (RQFE), in a weaker setting where all functions to be registered are known during setup. This weaker form of RQFE is nevertheless sufficient for RTT supporting an *unbounded* collusion of traitors. The resulting RTT scheme has a *transparent setup*, ciphertext size  $6\sqrt{L} + 4$  in number of group elements, and a *public tracing* algorithm. The scheme is based on prime-order groups and is *adaptively* secure in the generic (bilinear) group model (GGM).

(3) *Bounded-Collusion RTT.* We present an RFE for *linear* functions (RLF), in the ordinary setting where functions to be registered can be adaptively chosen after setup. This scheme is sufficient for RTT supporting a *bounded* collusion, where the maximum number of traitors is fixed at setup. We prove that our RLF is secure, against an arbitrary subset of *selectively* corrupted users, in the standard model. The security of this scheme rests upon a static  $q$ -type assumption, which we show to hold in the GGM. As a bonus, we further show how our RLF enables other new applications, such as registered threshold encryption (RTE) for  $t$ -out-of- $L$  thresholds, where ciphertexts are of size  $O(t)$  in number of group elements. RTE generalises the notion of distributed broadcast [9, 18, 31, 40] to  $t$ -out-of- $L$  thresholds. Our RLF, however, has a non-transparent setup. This yields structured crs in all our RLF applications.<sup>5</sup>

(4) *Prototype Implementation.* We provide an open-source prototype implementation of our RTT scheme with unbounded collusion. For a group of  $L = 1024$  users (which is currently the maximum size of a Whatsapp group chat), our benchmarks demonstrate that our scheme is quite practical: The key generation, encryption, and decryption algorithms take 553 ms, 51 ms, and 4 ms, respectively, whereas a ciphertext is of size 6.7 KB.

## 1.2 Related Work

*Traitor Tracing.* Traitor tracing was first introduced in [10], and ever since it has become one of the most studied topics in cryptography, with a large body of literature improving on the original proposal. Works on traitor tracing, e.g. [3,

<sup>4</sup> Supporting revocation is not our main focus in this work; we only discuss some revocation strategies informally.

<sup>5</sup> We note that one can use our RQFE to instantiate all these applications with a transparent setup.

7–9, 23–25, 30, 36, 41], focus on constructing schemes with sublinear<sup>6</sup> efficiency, in terms of size of public parameters and/or ciphertexts. This is possible by leveraging computational assumptions in bilinear groups or lattices.

To the best of our knowledge, essentially all prior traitor tracing schemes require a trusted setup to generate the public parameters and secret keys. An exception is a very recent work of Luo [33] that constructs a *Broadcast, Trace and Revoke* (or simply *Trace and Revoke*) system in the setting without a trusted authority, where each party samples its own keys. Trace and Revoke systems augment traitor-tracing with the ability to revoke decryption rights for any subset of users whose secret keys have been compromised. These systems have been extensively studied [3, 8, 14, 29, 34, 35] in the setting with a centrally trusted authority. [33] builds a Trace and Revoke scheme without any such central authority that also achieves essentially asymptotically optimal parameters, but relies on indistinguishability obfuscation (iO) [6, 28] and thus it is not concretely efficient. Though our work stems from a similar motivation, the model we consider is somewhat incomparable with that of [33]. In a nutshell, the key differences are as follows: (i) We allow for a (*preferably transparent*) setup outputting a common reference string, whereas [33] does not, (ii) we require *compact* master public keys, whereas the size of mpk in [33] is linear in the number of users, and (iii) the decryption in [33] requires random access to public keys and ciphertext, unlike ours that has a *short, deterministically computed* helper secret key (hsk). These differences in the model allow us to give concretely-efficient pairing-based constructions, instead of relying on the heavy machinery of iO.

Besides all the above, a series of work has explored the related notion of *distributed broadcast encryption* [9, 18, 31, 40], which does not concern traceability.

*Registered Cryptography.* Registered cryptography is a paradigm introduced by Garg et al. [19, 20] to remove the key-escrow from advanced forms of encryption that require a trusted setup. The paradigm has recently gained attention and a series of works have improved its functionality [26] and efficiency [11], ultimately leading to practical constructions from pairings [16, 22] and lattices [15]. The notion of registration-based encryption (RBE) was recently extended to the settings of attribute-based [27, 43] and functional encryption [12, 17].

*Concurrent Work.* Concurrently with our work, two other papers [13, 42] make independent progress on registered functional encryption. We discuss these works next.

First, [13] (a revised version of [12]) presents an RLFE scheme (with a *non-transparent* setup) from bilinear pairings and proves it secure in the GGM. On the other hand, [42] improves state of the art by presenting new RLFE and RQFE schemes from bilinear pairings, and proving them secure under different variants

---

<sup>6</sup> There exists a “trivial” traitor-tracing scheme where each party samples a public-key encryption individually, the master public key consists of the concatenation of the  $L$  public keys, and the ciphertext is simply the concatenation of encryptions under each individual key.

of standard  $k$ -Lin assumption. In particular, their RQFE and RLFE schemes satisfy very-selective simulation-based and adaptive indistinguishability-based security respectively. We note that the registered functional encryption schemes from [42] can be used to instantiate our main results in the standard model with different security trade-offs. However, their constructions suffer from the drawback of having *non-transparent* setup (i.e., their schemes have *structured* crs). We emphasize that one of our central aims in this work is to build an *unbounded-collusion* RTT with a *transparent* setup that is also *concretely* efficient. Additionally, our RLFE scheme with non-transparent setup (that we use to instantiate the bounded-collusion RTT), is proven *selectively* secure under a new parametrized assumption which we prove to hold in GGM. We note that, compared to [13,42], our RLFE scheme achieves comparable asymptotic (or, in some cases, better concrete) parameters. In particular, Table 1 compares the concrete parameters of our RLFE scheme with that of the concurrent works [13,42].

**Table 1.** Comparing concrete parameters (in terms of *total number of group elements*) of our slotted RLFE scheme with the same from concurrent works [13,42].  $L$  and  $n$  denote the number of slots and vector length of RLFE respectively. The notation  $d\text{-}(\mathbb{Z}_p)$  denotes  $d$  elements of  $\mathbb{Z}_p$ . The figures for RLFE from [42] are obtained by setting  $k = 2$  in the  $k$ -Lin assumption. For  $k > 2$ , the scheme of [42] would base on weaker assumptions, but at the cost of a strictly increasing number of group elements. For simplicity, we avoid accounting all extra group elements due to QA-NIZK from [42].

	crs	$\text{pk}_\ell$	$\text{sk}_\ell$	mpk	$\text{hsk}_\ell$	ct
RLFE [13]	$(n+1)L^2 + n + 2L + 1$	$L + 1$	$1\text{-}(\mathbb{Z}_p)$	$n + 2$	1	$n + 3$
RLFE [42]	$25nL^2 - 15nL + 35L + 20$	$5L + 35$	$25\text{-}(\mathbb{Z}_p)$	$10n + 30$	$5n + 10$	$5n + 15$
RLFE (Sect. 4.3)	$nL^2 + L$	$nL$	$n\text{-}(\mathbb{Z}_p)$	$n + 1$	$n + 2$	$n + 2$

### 1.3 Discussion

*Interactive vs Non-interactive Solution.* An alternative (generic) solution to remove trusted authorities in traitor-tracing systems, or in general any cryptographic systems, is to let participants *simulate* the trusted authority with a secure multi-party computation (MPC) protocol: All users run an MPC where they jointly sample the master secret key, and the output of each user consist of its tracing key, as well as the master public parameters. While this solution effectively bypasses the need for a trusted authority, it is undesirable for several reasons: (i) All users must be simultaneously online to run the MPC. Even a single user failing would stall the entire process. As the number of users in the system grows, this solution scales poorly. (ii) It is harder to support dynamic



joins of new users, since for every new user that joins a new MPC protocol must be jointly run by all participants. (iii) MPC protocols require interaction, which adds latency to the key registration process. (iv) Running the key generation of a traitor-tracing scheme as an MPC is computationally intense, making this solution computationally very expensive.

In this work we focus on the *non-interactive* settings, where users sample their own keys locally, and simply upload it to a public bulletin board once they are done. Different users do not even have to be aware of each other’s existence, and it is much easier to support a dynamic set of participants (more discussion on this later). For these reasons, we believe that the non-interactive setting is both theoretically more elegant and preferable from a practical standpoint.

*Common Reference String vs Trusted Authority.* We acknowledge that, in line with the literature on registered/distributed cryptography, our schemes are in the common reference string model, where all parties are assumed to receive a common reference string (CRS) that was sampled in a trusted manner. However, we highlight the fact that our RQFE scheme has a *transparent setup*, meaning that the CRS is just a collection of random bits.<sup>7</sup> In practice, this is desirable since one can substitute the CRS with a fixed seed for a hash function, which is then used to *obviously* compute the required group elements.<sup>8</sup> On the other hand, our RLFE has a structured setup. We claim that, even for the case of a non-transparent setup, this model is substantially better than having a trusted authority, since there is no long-term secret that needs to be stored. It also resolves questions about the availability of the trusted authority, and how parties can establish secure communication channels for receiving their keys.

*Static Joins vs Dynamic Joins.* Throughout this work, we will always assume that the set of users that register their keys is fixed ahead of time, and the public parameters are aggregated only after all users have registered their keys. That is, we assume that the set of users participating in the protocol is *static*, and if a new user joins the system, the master public key needs to be recomputed and all users have to be notified of this change, and (possibly) must update their information. [27] refers to this as the *slotted* setting.

In practice, it is desirable to allow users to join the system dynamically, and one does not want to re-initialise the public parameters and/or to notify all existing users. Fortunately, it is possible to generically move from the slotted/static settings to support dynamic joins, while minimising the number of updates. Informally, the transformation works by partitioning the users in sets of exponentially increasing cardinality, e.g.  $\{S_i : |S_i| = 2^i\}_{i \in [\log(L)]}$ , and filling those sets as users join, starting from the smaller ones. Updates then only need to be issued when a set is filled up and needs to be transferred to the next empty set. It is easy to see that each user receives at most  $\log(L)$  updates throughout

---

<sup>7</sup> Looking ahead, this immediately endows our final RTT with a transparent setup.

<sup>8</sup> Hashing into groups with a bilinear pairing is a well-studied problem in the literature, see e.g. [38].

its lifetime. Variants of this transformation have been described many times in the literature [17, 19, 20, 22, 27, 31] and we refer the reader to these works for more details. In what follows, we will only describe schemes in the slotted/static settings, with the understanding that dynamic joins can be supported with this transformation.

## 2 Technical Overview

We highlight the technical innovations of our work. We begin by showing how constructing RTT boils down to building the right notion of RQFE, then we present our RFE schemes. We conclude by outlining registered threshold encryption as another new application of RFEs.

### 2.1 Registered Traitor-Tracing

To set some context, let us make more concrete the desiderata for an RTT scheme. In an RTT scheme, the setup outputs a (preferably *unstructured*) common reference string  $\text{crs}$ . Each party  $i$  starts by generating its own pair of public and secret keys  $(\text{pk}_i, \text{sk}_i)$  relative to  $\text{crs}$ . Upon collecting all the public keys  $(\text{pk}_1, \dots, \text{pk}_L)$ , *anyone* can use the  $\text{crs}$  to *deterministically* compute a short master public key  $\text{mpk}$  and the helper decryption key  $\text{hsk}_i$  for each user  $i$ . Note that  $\text{hsk}_i$  is *not* a replacement for the user secret key, but rather an additional (*publicly computable*) information needed to complete decryption. Given  $\text{mpk}$ , anyone can encrypt  $m$  in such a way that only a registered user  $i$  is able to obtain the message, using its secret key  $\text{sk}_i$  and helper key  $\text{hsk}_i$ . Additionally, RTT should fulfill a strong traceability property: If a malicious user  $i^*$  builds a decryption box  $D$  (which receives ciphertexts and outputs the corresponding message with non-negligible probability), then the user  $i^*$  can be caught given only black-box access to  $D$ . The RTT scheme is said to be bounded-collusion secure if the setup additionally inputs the maximum number of traitors, else it supports an unbounded collusion.

*TT via Functional Encryption.* To better understand the challenge of constructing (R)TT, it is useful to recall how to construct traditional traitor-tracing schemes (with a trusted authority). The work of Boneh, Sahai, and Waters [7] reduces this problem to a simpler cryptographic primitive called private linear broadcast encryption (PLBE) and shows how to generically turn a PLBE scheme into a traitor-tracing scheme. In a nutshell, a PLBE is a broadcast encryption scheme with an additional trace-encrypt algorithm. This algorithm takes as input an index  $i \in [L]$  and a message, and generates an ordinary-looking ciphertext of the message which can only be decrypted by user  $\ell \geq i$ . Importantly, this ciphertext must keep the index  $i$  hidden (except to users  $i$  and  $i + 1$ , who can trivially test the position of the index). The trace-encrypt algorithm can be used in a linear scan to identify the user with the smallest index who contributed to creating a rogue decryption device.

Abstracting even further, it turns out that PLBE is nothing but a special case of functional encryption (FE) [21], where keys are associated with an index  $\ell$  and a predicate  $F_\ell(i, m)$  such that

$$F_\ell(i, m) := \begin{cases} m & \text{if } i \leq \ell \\ 0 & \text{otherwise} \end{cases}$$

whereas ciphertexts contain information about the message  $m$  and the index  $i$ . This connection is made explicit in [21] which shows that an FE for quadratic functions (QFE) is sufficient to implement the above comparison predicate, and consequently PLBE, with ciphertext size  $O(\sqrt{L})$ . Thus, the problem of traitor tracing is nothing but QFE in disguise.<sup>9</sup>

For the (weaker) bounded-collusion setting, Agrawal et al. [2] show how to reduce the problem of traitor-tracing (with revocation) to that of bounded-collusion FE for linear functions (LFE).

*Registered FE: Removing the Authority.* Via the aforementioned series of transformations, we have reduced the task of constructing traitor-tracing without authority to that of constructing QFE/LFE without authority. This notion was recently introduced under the name of *registered functional encryption* (RFE) [12, 17] as a natural generalisation of registration-based encryption [19]. In short, RFE provides a mechanism to publicly aggregate  $L$  independent key-function tuples  $(\text{pk}_1, f_1), \dots, (\text{pk}_L, f_L)$  into a digest, so that a ciphertext of  $m$  generated with respect to the digest can be decrypted by  $\text{sk}_j$  to recover  $f_j(m)$ .

For the remainder of this overview, we will focus on describing our RFE schemes (for quadratic and linear functions), along with other applications. Extending the transformation from QFE/LFE to traitor-tracing in the registered settings require some care, but the main ideas are analogous to the traditional settings. Therefore, we omit them here and refer the reader to Sect. 5 for more details.

## 2.2 RQFE in the GGM

Our first observation that facilitates our task is that one does not need the full power of (R)QFE to build traitor tracing. Since the functions  $f_1, \dots, f_L$  depend only on the identity of each user, it suffices to build a scheme where all functions associated with secret keys are known ahead of time. In other words, we can assume that each user knows all the other functions during key generation. With this observation in mind, we describe our RQFE below.

Conceptually, we build our RQFE by compiling a traditional QFE into a registered one, provided that it satisfies a *master secret key homomorphism*. In

<sup>9</sup> Note that linearising a quadratic polynomial achieves the desired functionality, but nullifies the efficiency of the transformation. In particular, the resulting PLBE scheme would have ciphertexts linear in  $L$ , which does not improve over trivial constructions.

other words, we want the master public key of the scheme to be some encoding of the master secret key, that satisfies the following homomorphic relation:

$$\underbrace{\text{Encode}(\text{msk}_0)}_{\text{mpk}_0} * \underbrace{\text{Encode}(\text{msk}_1)}_{\text{mpk}_1} = \text{Encode}(\text{msk}_0 + \text{msk}_1)$$

and furthermore for all functions  $f$ :

$$\underbrace{\text{KGen}(\text{msk}_0, f)}_{\text{sk}_f^{(0)}} * \underbrace{\text{KGen}(\text{msk}_1, f)}_{\text{sk}_f^{(1)}} = \text{KGen}(\text{msk}_0 + \text{msk}_1, f).$$

The exact specifications of the encoding function  $\text{Encode}$  and the group operation  $*$  are irrelevant for this explanation. To define the master public key of the scheme, each user samples a local key pair  $(\text{mpk}_i, \text{msk}_i)$  and we define the *global* master public key as

$$\widetilde{\text{mpk}} = \text{mpk}_1 * \dots * \text{mpk}_L = \text{Encode}(\text{msk}_1 + \dots + \text{msk}_L)$$

which can be computed publicly using the master public keys published by each user. In effect, the  $L$  users are sharing (in the sense of additive secret-sharing) the master secret key of the new combined key  $\widetilde{\text{mpk}}$ . The users will then also publish enough information to help the  $i$ -th user computing a functional secret key under the new master public key. Here is where we leverage the fact that all functions are known in advance, and we ask each user to publish, along with their  $\text{mpk}_j$  all functional keys, except for their own function. In other words, the  $j$ -th user also outputs

$$\left\{ \text{sk}_{f_i}^{(j)} = \text{KGen}(\text{msk}_j, f_i) \right\}_{i \neq j}.$$

Arranging all of these public information in matrix form, and applying the homomorphic operator row-wise, we obtain:

$$\begin{pmatrix} \downarrow & \text{sk}_{f_2}^{(1)} & \dots & \text{sk}_{f_L}^{(1)} \\ \text{sk}_{f_1}^{(2)} & \downarrow & \dots & \text{sk}_{f_L}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \text{sk}_{f_1}^{(L)} & \text{sk}_{f_2}^{(L)} & \dots & \downarrow \end{pmatrix} \xrightarrow{*} \begin{pmatrix} \text{KGen}(\sum_{j \neq 1} \text{msk}_j, f_1) \\ \text{KGen}(\sum_{j \neq 2} \text{msk}_j, f_2) \\ \vdots \\ \text{KGen}(\sum_{j \neq L} \text{msk}_j, f_L) \end{pmatrix}$$

Note that the  $i$ -th combined key is almost a valid functional secret key for  $f_i$  under  $\widetilde{\text{mpk}}$ , except that it is missing the contribution of  $\text{msk}_i$ . However, the  $i$ -th user is the one that sampled  $\text{msk}_i$  in the first place, and therefore it can easily fill the missing value to obtain

$$\begin{aligned} \widetilde{\text{sk}}_{f_i} &= \text{KGen} \left( \sum_{j \neq i} \text{msk}_j, f_i \right) * \text{KGen}(\text{msk}_i, f_i) \\ &= \text{KGen} \left( \sum_j \text{msk}_j, f_i \right). \end{aligned}$$

At this point, decryption and encryption correctness simply follow by the correctness of the original FE scheme, except that we now have substituted the key authority with a fully distributed setup.

*Instantiating the Transformation.* Given this general template outlined above, all that is left is to look into the literature of traditional QFE schemes, and find a compatible one. It turns out that a handful of schemes satisfy this homomorphic property. However, while all schemes obtained via this transform are correct, not all of them can be proven secure while having a transparent setup. For instance, the RQFE scheme obtained by transforming the QFE of Wee [39] is unfortunately broken due to linear attacks.<sup>10</sup> The only QFE scheme which we are aware of that survives the transformation (with a transparent setup) is that of Baltico et al. [5], which was only proven to be secure in the GGM. Consequently, our RQFE inherits the security in the GGM. Proving security of this template turns out to be a nuanced task, since we have to deal with potentially malformed keys<sup>11</sup> and adaptive corruption queries. We refer to Sect. 4.2 for more technical details.

### 2.3 RLFE in the Standard Model

Our RFE construction for linear functions is conceptually similar to the recent works of [27, 43] on registered attribute-based encryption (RABE) and can be summarised by the following idea. Starting with a base FE scheme, the major challenge is to construct a one-user RFE which is correct and secure. Given this, we can construct an  $L$ -user RFE by running  $L$  parallel instances of the one-user RFE, where the digest  $\text{mpk}$  aggregates all individual  $\text{mpk}_i$  from the  $L$  instances. To ensure decryption correctness, the  $\text{Aggr}$  algorithm outputs helper keys which correspond to the cross-terms due to  $(\text{mpk}_j)_{j \neq i}$  in  $\text{mpk}$  for each user  $i$ .

*Our RLFE.* While the above general strategy can be applied to various existing linear FEs, adapting their security proofs to the registered setting is tricky. We settle at basing on the scheme of [1] due to its simplicity, which we recall:

$$\text{mpk} = [\mathbf{w}^T], \quad \text{msk} = \mathbf{w}^T, \quad \text{sk}_y = \mathbf{w}^T \mathbf{y}, \quad \text{ct}_x = ([s], [s\mathbf{w}^T + \mathbf{x}^T])$$

<sup>10</sup> We note that the concurrent work of [42] builds pairing-based RQFE in the standard model following [39] techniques, which is a different approach than ours. Notably, their RQFE has a large, structured  $\text{crs}$  that was also required to program to prove security, thus bypassing the possibility of having a transparent setup.

<sup>11</sup> In RFE, an adversary can register its own (potentially malformed) keys. We handle security against such keys in various generic and non-generic ways for our RFE schemes. Specifically, the generic technique employs NIZK which works for any RFE scheme, whereas we also show tailor-made techniques that work for our RQFE scheme. However, we avoid discussing those here for readability. We refer to Remark 1 and subsequent detailed discussions on this.

for some  $\mathbf{w}, \mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^n$  and  $s \in \mathbb{Z}_p$ , where  $[\cdot]$  represents component-wise exponentiations in some respective group. To decrypt, compute

$$[s\mathbf{w}^T + \mathbf{x}^T]\mathbf{y} - [s]\mathbf{w}^T\mathbf{y} = [\mathbf{x}^T\mathbf{y}].$$

We turn this into a one-user RLFE for a given function  $\mathbf{y}$  with the following steps. Fix  $[\mathbf{w}^T]$  in the  $\text{crs}$  and let  $\text{mpk}' = [\mathbf{w}^T\mathbf{y}]$ . Correspondingly, let  $\text{ct}_{\mathbf{x}} = ([s\mathbf{w}^T\mathbf{y}], [s\mathbf{w}^T + \mathbf{x}^T])$ , so that the same decryption equation (and both correctness and security of the base scheme) applies:

$$[s\mathbf{w}^T + \mathbf{x}^T]\mathbf{y} - [s\mathbf{w}^T\mathbf{y}] = [\mathbf{x}^T\mathbf{y}].$$

Crucially,  $\text{mpk}'$  “Pedersen-commits” the function  $\mathbf{y}$  using the “key”  $\mathbf{w}^T$ , which is then inherited in  $\text{ct}$ , so that no one can decrypt to  $\mathbf{x}^T\mathbf{y}'$  for  $\mathbf{y}' \neq \mathbf{y}$ . This yields a “public RLFE” that anyone can decrypt via the above equation, and to make this available only to the registered user, the idea is to (additively) secret-share the commitment key. We let  $\text{pk} = [\mathbf{v}]$ ,  $\text{sk} = \mathbf{v}$ , and the new commitment key be  $\mathbf{w} + \mathbf{v}$ , shared by  $\text{crs}$  and the user. The resulting one-user RLFE has

$$\begin{aligned} \text{crs} &= [\mathbf{w}^T], \quad \text{mpk} = ([\mathbf{w}^T], [(\mathbf{w}^T + \mathbf{v}^T)\mathbf{y}]), \quad \text{pk} = [\mathbf{v}^T], \\ \text{sk} &= \mathbf{v}^T, \quad \text{ct}_{\mathbf{x}} = ([s], [s(\mathbf{w}^T + \mathbf{v}^T)\mathbf{y}], [s\mathbf{w}^T + \mathbf{x}^T]) \end{aligned}$$

and decryption follows from

$$[s\mathbf{w}^T + \mathbf{x}^T]\mathbf{y} + [s]\mathbf{v}^T\mathbf{y} - [s(\mathbf{w}^T + \mathbf{v}^T)\mathbf{y}] = [\mathbf{x}^T\mathbf{y}].$$

In a nutshell, security follows from two facts: Only the user who knows the share  $\mathbf{v}$  can access the “public RLFE”; furthermore decrypting to only  $\mathbf{x}^T\mathbf{y}$  is safeguarded by the other share  $\mathbf{w}$ . From here, we apply the  $L$ -parallel-instances compiler to obtain an  $L$ -user RLFE. To prevent mix-and-match of helper keys, i.e. cross-terms across the  $L$  instances, a randomisation factor for each user is introduced and bound to their helper keys, which is done via pairing. Our final RLFE has a non-transparent setup. The scheme of [1] is proven from DDH with selective-security. Our scheme also inherits the same security and the randomisation in helper keys lead to our  $q$ -type assumption (for  $q = L$  number of users), which is essentially a  $q$ -type variant of DDH generalised into the pairing setting.<sup>12</sup> We present our full RLFE construction in Sect. 4.3 and show in the full version, how this yields a single-key RFE for all circuits.

## 2.4 Registered Threshold Encryption

As one of the bonus applications, we discuss how RLFE helps in removing the trusted setup in threshold encryption. In other words, we show how to build

<sup>12</sup> We note that the concurrent work in [42] improve the state of the art by building an adaptively secure RLFE scheme from a static assumption on bilinear maps in the standard model.

*registered threshold encryption* (RTE). Recall that, in traditional threshold encryption, the public parameters of the system are generated together with  $L$  users' secret keys. Given an encryption  $\text{ct}$  of a message  $m$ , each user can compute partial decryption shares using its secret key. Once we have  $t$  partial decryption shares, where the recovery threshold  $t \leq L$  is specified in the public parameters, the message  $m$  can be recovered. In terms of security, we want that an adversary holding less than  $t$  secret keys is unable to break semantic security of the scheme. In RTE, parties generate their own public keys and these are later aggregated into a short master public key. The system should preserve the “threshold decryption” functionality as in traditional threshold encryption.

To compile an RLFE into an RTE, each party  $i$  simply runs the RLFE key generation on a vector  $\mathbf{i} = (1, i, \dots, i^{t-1}) \in \mathbb{Z}_p^t$ . To encrypt a message  $m \in \{0, 1\}$ , the encryptor first performs Shamir secret sharing, i.e. sampling a random degree  $(t - 1)$  polynomial  $P$  over  $\mathbb{Z}_p$  such that  $P(0) = m$ . Let  $\mathbf{p} \in \mathbb{Z}_p^t$  be the coefficient vector of  $P$ . The encryptor encrypts  $\mathbf{p}$  using the underlying RLFE scheme. By the security of the RLFE, a party holding a secret key  $\text{sk}_i$  learns  $\langle \mathbf{i}, \mathbf{p} \rangle = P(i)$  and nothing else about the polynomial  $P$ . Once we have  $t$  different evaluations of the polynomial, we can recover  $P(0) = m$  by Lagrange interpolation. Since our RLFE has a non-transparent setup, this is also inherited by our RTE. However, we can use our RQFE to instantiate the RTE with a transparent setup. In the full version, we sketch how this yields a distributed broadcast encryption with transparent setup. In the full version we detail our RTE construction.

One subtle issue that we omitted so far is that the RLFE decryption actually allows a party  $i$  to recover the inner product  $\langle \mathbf{i}, \mathbf{p} \rangle = P(i)$  *in the exponent* of a target group  $\mathbb{G}_T$  element  $[P(i)]_T$  from the underlying bilinear pairing. This does not create an issue as Lagrange interpolation is a linear function and thus, we can perform it in the exponent to recover  $[P(0)]_T$ . Since  $P(0) = m \in \{0, 1\}$ , we can brute-force  $m$  from  $[m]_T$ .

### 3 Preliminaries

*Notation.* We denote the security parameter by  $\lambda \in \mathbb{N}$  throughout this paper and assume it as an implicit input to all algorithms. We write  $[n] = \{1, \dots, n\}$  and  $[0, n] = \{0\} \cup [n]$  for any  $n \in \mathbb{N}$ . Capital and small bold-face letters (like  $\mathbf{M}$  and  $\mathbf{x}$ ) denote matrices and (column) vectors respectively. Capital and small letters (such as  $S$  and  $x$ ) in general denote sets and concrete algebraic variables respectively (with any exceptions being stated explicitly). A tuple  $T = (t_i)_{i \in [n]}$  defines an ordered set with elements indexed from  $[n]$  for any  $n \in \mathbb{N}$ . Accordingly,  $|S|$  and  $|\mathbf{x}|$  respectively denotes the cardinality of set  $S$  and the length of a vector  $\mathbf{x}$ . We write  $x \leftarrow_s X$  to denote sampling an element  $x$  from  $X$  uniformly at random. We write  $\mathcal{A}$  for a probabilistic polynomial time (PPT) adversary that runs in time polynomial in  $\lambda$ . A function in  $\lambda$ , denoted by  $\text{negl}(\lambda) : \mathbb{N} \mapsto \mathbb{R}$ , is called negligible if it vanishes faster than the inverse of any polynomial in  $\lambda$ , i.e.  $\text{negl}(\lambda) \in \mathcal{O}(1/p(\lambda))$  for all positive polynomials  $p(\lambda)$ .

*Prime-Order Bilinear Groups.* Throughout this work, we use cyclic groups of prime order  $p$  with an asymmetric bilinear map endowed on them. We assume a PPT bilinear group generator algorithm  $\text{GGen}$  that takes  $\lambda \in \mathbb{N}$  as input and outputs  $\mathcal{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ , where  $p$  is a prime of  $\Theta(\lambda)$  bits,  $\mathbb{G}_1 = \langle g_1 \rangle$ ,  $\mathbb{G}_2 = \langle g_2 \rangle$ ,  $\mathbb{G}_T = \langle g_T \rangle = \langle e(g_1, g_2) \rangle$  are cyclic groups of order  $p$  with  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$  being a non-degenerate bilinear map. We use the implicit (bracket) notation for group elements: for  $\mathbf{M}, \mathbf{M}' \in \mathbb{Z}_p^{k_1 \times k_2}$ , define  $[\mathbf{M}]_{\mathbf{t}} = g_{\mathbf{t}}^{\mathbf{M}} := (g_{\mathbf{t}}^{m_{i,j}})$  and  $[\mathbf{M}]_{\mathbf{t}} + [\mathbf{M}']_{\mathbf{t}} := [\mathbf{M} + \mathbf{M}' \bmod p]_{\mathbf{t}}$  for  $\mathbf{t} \in \{1, 2, T\}$  and  $k_1, k_2 \in \mathbb{N}$ . We also denote  $[1]_1 := g_1$ ,  $[1]_2 := g_2$ , and abbreviate “ $e$ ” with “ $\cdot$ ”, i.e. for matrices  $\mathbf{M}_1, \mathbf{M}_2$  of appropriate dimensions,  $e([\mathbf{M}_1]_1, [\mathbf{M}_2]_2)$  is written as  $[\mathbf{M}_1]_1 [\mathbf{M}_2]_2 = [\mathbf{M}_1 \mathbf{M}_2]_T = g_T^{\mathbf{M}_1 \mathbf{M}_2}$ . We express sampling a bilinear group instance as  $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, [1]_1, [1]_2, \cdot) \leftarrow \text{GGen}(1^\lambda)$ .

## 4 Registered Functional Encryption

We define and construct the core building blocks for our applications, namely registered functional encryption (RFE) for quadratic and linear functions. In particular, we define the syntax and security of RFE in Sect. 4.1. Then, Sects. 4.2 and 4.3 provide schemes for weak RFE and RFE for quadratic and linear functions respectively. Both our RFE schemes are proven secure in presence of well-formed keys. The full version describes generic and concrete ways of tackling malicious keys to transcend this limitation.

### 4.1 Definitions

We define RFE and a variant which we call weak RFE. The main difference between the two is that, in the weak variant, the set of functions to be registered is known already at setup time. Below we primarily define RFE and describe the difference of the weak variant inline.

**Definition 1 (Registered Functional Encryption).** *A registered functional encryption (RFE) scheme for message space  $\mathcal{M}$ , ciphertext space  $\mathcal{C}$ , function class  $\mathcal{F}$  and number of users  $L$  consists of the following tuple of PPT algorithms (Setup, KGen, Aggr, Enc, Dec):*

- $\text{crs} \leftarrow \text{Setup}(1^\lambda)$ : On input the security parameter  $1^\lambda$ , the setup algorithm outputs a common reference string  $\text{crs}$ .
- $(\text{pk}_\ell, \text{sk}_\ell) \leftarrow \text{KGen}(\text{crs}, \ell \in [L])$ : The key generation algorithm outputs a pair of public and secret keys  $(\text{pk}_\ell, \text{sk}_\ell)$  for user  $\ell$ .
- $(\text{mpk}, (\text{hsk}_\ell)_{\ell \in [L]}) \leftarrow \text{Aggr}(\text{crs}, (\text{pk}_\ell, f_\ell)_{\ell \in [L]})$ : On input  $\text{crs}$  and the tuple of public key  $\text{pk}_\ell$  and function  $f_\ell \in \mathcal{F}$  of all users  $\ell \in [L]$ , the deterministic aggregation algorithm outputs a master public key  $\text{mpk}$  and a tuple of helper secret keys  $(\text{hsk}_\ell)_{\ell \in [L]}$ .
- $\text{ct} \leftarrow \text{Enc}(\text{mpk}, \mu)$ : On input  $\text{mpk}$  and a message  $\mu \in \mathcal{M}$ , the encryption algorithm outputs a ciphertext  $\text{ct} \in \mathcal{C}$ .



- $\mu' \leftarrow \text{Dec}(\text{sk}_\ell, \text{hsk}_\ell, \text{ct})$ : On input a ciphertext  $\text{ct}$  together with a secret key  $\text{sk}_\ell$  and a helper secret key  $\text{hsk}_\ell$ , the decryption algorithm outputs  $\mu'$ .

A weak RFE has the same syntax as an RFE, except that the tuple of functions  $(f_\ell)_{\ell \in [L]}$  is input to  $\text{Setup}$  instead of to  $\text{Aggr}$ .

**Definition 2 (Correctness).** An RFE scheme is said to be correct, if for all  $\lambda \in \mathbb{N}$ ,  $L \in \text{poly}$ ,  $\mu \in \mathcal{M}$ ,  $k \in [L]$ ,  $(f_\ell)_{\ell \in [L]} \in \mathcal{F}^L$ ,  $\text{crs} \in \text{Setup}(1^\lambda)$ ,  $(\text{pk}_k, \text{sk}_k) \in \text{KGen}(\text{crs}, k)$ , it holds that

$$\Pr \left[ \begin{array}{l} \mu' = f_k(\mu) \\ \text{ct} \leftarrow \text{Enc}(\text{mpk}, \mu) \\ \mu' \leftarrow \text{Dec}(\text{sk}_k, \text{hsk}_k, \text{ct}) \end{array} \middle| \begin{array}{l} (\text{mpk}, (\text{hsk}_\ell)_{\ell \in [L]}) \leftarrow \text{Aggr}(\text{crs}, (\text{pk}_\ell, f_\ell)_{\ell \in [L]}) \end{array} \right] = 1.$$

Correctness of a weak RFE is defined analogously with the only differences being that  $(f_\ell)_{\ell \in [L]}$  is input to  $\text{Setup}$  instead of to  $\text{Aggr}$ .

**Definition 3 (Strong Compactness).** An RFE is said to be strongly compact, if for all  $\lambda \in \mathbb{N}$ ,  $L \in \text{poly}$ ,  $(f_\ell)_{\ell \in [L]} \in \mathcal{F}^L$ ,  $\text{crs} \in \text{Setup}(1^\lambda)$ ,  $(\text{pk}_\ell, \text{sk}_\ell) \in \text{KGen}(\text{crs}, \ell)$ , and  $(\text{mpk}, (\text{hsk}_\ell)_{\ell \in [L]}) \in \text{Aggr}(\text{crs}, (\text{pk}_\ell, f_\ell)_{\ell \in [L]})$ , it holds that  $|\text{mpk}|$ ,  $|\text{hsk}_\ell|$ ,  $|\text{ct}|$  are of size  $\text{poly}[\lambda, \log L]$ .<sup>13</sup> Strong compact weak RFEs are defined analogously.

**Definition 4 (Security).** An RFE scheme  $\Pi$  is said to be secure, if for any PPT  $\mathcal{A}$  it holds that

$$\left| \Pr [\text{Exp}_{\Pi, \mathcal{A}}^0(1^\lambda) = 1] - \Pr [\text{Exp}_{\Pi, \mathcal{A}}^1(1^\lambda) = 1] \right| \leq \text{negl}(\lambda),$$

where  $\text{Exp}_{\Pi, \mathcal{A}}^b$  is defined in Fig. 1. The security of a weak RFE is defined analogously, with the only difference that  $(f_\ell)_{\ell \in [L]}$  is declared by  $\mathcal{A}$  upfront and input to  $\text{Setup}$  instead of to  $\text{Aggr}$ .

We also consider the notion of selective-security with static corruption, where the experiment is same as that in Fig. 1, except that  $\mathcal{A}$  declares the messages  $(\mu_0, \mu_1)$  and the set of corrupt users  $C \subseteq [L]$  at the beginning of the experiment (i.e. the corruption oracle  $\text{CorrO}$  is withheld from  $\mathcal{A}$ ).

*Remark 1 (On Malicious Keys and Key Queries).* In Definition 4 we require  $\mathcal{A}$  to output the randomness  $(\mathbf{r}_\ell)_{\ell \in M}$  for keys generated by  $\mathcal{A}$ , the setting which our RFEs will be proven secure. We defer handling malicious keys without this requirement to the full version, where the relevant  $\text{IsValid}$  algorithm and completeness property are also introduced. For simplicity we only allow a single key query per user  $\ell$ . The mildly stronger notion of allowing multiple key queries per user is implied so long as  $\text{KGen}$  is stateless (so that the same reduction still applies when simulating multiple keys for the same user) and holds true for both of our RFE schemes.

<sup>13</sup> Our definition is stronger than existing RFE compactness [17], since it additionally requires *succinct ciphertexts*. We note that the concurrent work of [42] also specifies this property in their definition.

$\text{Exp}_{II, \mathcal{A}}^b(1^\lambda)$	$\text{KGenO}(\ell)$
$\text{crs} \leftarrow \text{Setup}(1^\lambda)$	<b>if</b> $K[\ell] = \perp$
$(\mu_0, \mu_1, (\text{pk}_\ell, f_\ell)_{\ell \in [L]}, (\mathbf{r}_\ell)_{\ell \in M}) \leftarrow \mathcal{A}^{\text{CorrO}(\cdot), \text{KGenO}(\cdot)}(\text{crs})$	$(\text{pk}_\ell, \text{sk}_\ell) \leftarrow \text{KGen}(\text{crs}, \ell)$
// $\mathcal{A}$ provides randomness for set $M$ of maliciously generated keys	$K[\ell] := (\text{pk}_\ell, \text{sk}_\ell)$
<b>assert</b> $[L] \setminus M \subseteq K \subseteq [L]$	$(\text{pk}_\ell, \text{sk}_\ell) \leftarrow K[\ell]$
<b>assert</b> $\text{pk}_\ell \in \text{KGen}(\text{crs}, \ell; \mathbf{r}_\ell) \quad \forall \ell \in M$	<b>return</b> $\text{pk}_\ell$
<b>assert</b> $f_\ell(\mu_0) = f_\ell(\mu_1) \quad \forall \ell \in C \cup M$	<hr/>
$(\text{mpk}, (\text{hsk}_\ell)_{\ell \in [L]}) \leftarrow \text{Aggr}(\text{crs}, (\text{pk}_\ell, f_\ell)_{\ell \in [L]})$	$\text{CorrO}(\ell)$
$\text{ct}^* \leftarrow \text{Enc}(\text{mpk}, \mu_b)$	$C := C \cup \{\ell\}$
$b' \leftarrow \mathcal{A}(\text{ct}^*)$	$(\text{pk}_\ell, \text{sk}_\ell) \leftarrow K[\ell]$
<b>return</b> $b'$	<b>return</b> $\text{sk}_\ell$

Fig. 1. Security experiment for RFE.

## 4.2 Weak RFE for Quadratic Functions

We build a weak RFE scheme for quadratic functions with a *transparent* setup in Fig. 2, i.e. the  $\text{crs}$  is constructed with public randomness.

Let  $n_1, n_2, L \in \text{poly}$ . For any  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, [1]_1, [1]_2, \cdot)$  output by  $\text{GGen}(1^\lambda)$ , we construct an RFE for the message space  $\mathcal{M} = \mathbb{Z}_p^{n_1} \times \mathbb{Z}_p^{n_2}$ , the class of quadratic functions  $\mathcal{F}$  being

$$\{(f : \mathcal{M} \rightarrow [\mathbb{Z}_p]_T, f(\mathbf{x}, \mathbf{y}) \mapsto [\mathbf{x}^T \mathbf{F} \mathbf{y} \bmod p]_T) : \mathbf{F} \in \mathbb{Z}_p^{n_1 \times n_2}\},$$

and (an upper bound of)  $L$  number of users. Since for any  $f \in \mathcal{F}$ ,  $f(\mathbf{x}, \mathbf{y}) \mapsto [\mathbf{x}^T \mathbf{F} \mathbf{y} \bmod p]_T$  is fully described by  $\mathbf{F}$ ,  $\mathbb{G}_T$  and  $p$  whereas  $\mathbb{G}_T, p$  are publicly fixed, we simply write  $\mathbf{F}$  for such. Further, for any  $\ell \in \mathbb{N}$  and  $\mathbf{F}_\ell \in \mathcal{F}$  we denote its  $(i, j)$ -th entry as  $f_{i,j}^{(\ell)} \in \mathbb{Z}_p$ .

The analysis of the scheme can be found in the full version.

## 4.3 RFE for Linear Functions

Let  $n, L \in \text{poly}$ . For any  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, [1]_1, [1]_2, \cdot)$  output by  $\text{GGen}(1^\lambda)$ , we construct in Fig. 3 an RFE for the message space  $\mathcal{M} = \mathbb{Z}_p^n$ , the class of linear functions  $\mathcal{F}$  being

$$\{(f : \mathbb{Z}_p^n \rightarrow [\mathbb{Z}_p]_T, f(\mathbf{x}) \mapsto [\mathbf{x}^T \mathbf{y} \bmod p]_T) : \mathbf{y} \in \mathbb{Z}_p^n\},$$

and (an upper bound of)  $L$  users. Since any  $f \in \mathcal{F}$ ,  $f(\mathbf{x}) \mapsto [\mathbf{x}^T \mathbf{y} \bmod p]_T$  is fully described by  $\mathbf{y}$ ,  $\mathbb{G}_T$  and  $p$  whereas  $\mathbb{G}_T, p$  are publicly fixed, we simply write  $\mathbf{y}$  for such. Our RFE for linear functions has a non-transparent setup. We remark that the scheme can be trivially extended to one supporting the function class mapping to  $\mathbf{x}^T \mathbf{y} \bmod p$ , i.e. in plain instead of as target group element, with appropriate bound  $B$  on the image space, by letting the decryption algorithm solving for the discrete log solution.

<p><b>Setup</b>(<math>1^\lambda, (\mathbf{F}_\ell)_{\ell \in [L]}</math>)</p> <hr/> $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, [1]_1, [1]_2, \cdot) \leftarrow \text{GGen}(1^\lambda)$ <b>for</b> $\ell \in [L] : \gamma_\ell \leftarrow \mathbb{Z}_p$ $\mathbf{t} \leftarrow \mathbb{Z}_p^{n_2}$ <b>return</b> $\text{crs} := \left( \mathcal{G}, (\mathbf{F}_\ell)_{\ell \in [L]}, ([\gamma_\ell]_2)_{\ell \in [L]}, [\mathbf{t}]_2 \right)$	<p><b>KGen</b>(<math>\text{crs}, \ell</math>)</p> <hr/> $\mathbf{s}_\ell \leftarrow \mathbb{Z}_p^{n_1}; w_\ell \leftarrow \mathbb{Z}_p$ <b>for</b> $k \in [L] :$ $[\mathbf{dk}_{\ell,k}]_2 := [\mathbf{s}_\ell^\top \mathbf{F}_k \mathbf{t} + \gamma_k w_\ell]_2$ $\text{pk}_\ell := \left( [s_\ell]_1, [w_\ell]_1, ([\mathbf{dk}_{\ell,k}]_2)_{k \in [L] \setminus \{\ell\}} \right)$ $\text{sk}_\ell := [\mathbf{dk}_{\ell,\ell}]_2$ <b>return</b> $(\text{pk}_\ell, \text{sk}_\ell)$
<p><b>Enc</b>(<math>\text{mpk}, (\mathbf{x}, \mathbf{y})</math>)</p> <hr/> <b>parse</b> $[s]_1 = ([s_1]_1, \dots, [s_{n_1}]_1)$ <b>parse</b> $[t]_2 = ([t_1]_2, \dots, [t_{n_2}]_2)$ <b>parse</b> $(\mathbf{x}, \mathbf{y}) = (x_1, \dots, x_{n_1}, y_1, \dots, y_{n_2})$ $\alpha \leftarrow \mathbb{Z}_p$ $\mathbf{M} \leftarrow \text{GL}_2(\mathbb{Z}_p)$ $[C_1]_1 := [\alpha]_1$ $[C_2]_1 := [\alpha w]_1$ $[C_{3,i}]_1 := \left[ \mathbf{M}^{-1} \begin{pmatrix} x_i \\ \alpha s_i \end{pmatrix} \right]_1, \forall i \in [n_1]$ $[C_{4,j}]_2 := \left[ \mathbf{M} \cdot \begin{pmatrix} y_j \\ -t_j \end{pmatrix} \right]_2, \forall j \in [n_2]$ $\text{ct} := \left( [C_1]_1, [C_2]_1, ([C_{3,i}]_1)_{i \in [n_1]}, ([C_{4,j}]_2)_{j \in [n_2]} \right)$ <b>return</b> $\text{ct}$	<p><b>Aggr</b>(<math>\text{crs}, (\text{pk}_\ell)_{\ell \in [L]}</math>)</p> <hr/> $[s]_1 := \left[ \sum_{\ell \in [L]} \mathbf{s}_\ell \right]_1; [w]_1 := \left[ \sum_{\ell \in [L]} w_\ell \right]_1$ <b>for</b> $k \in [L] :$ $[h_{1,k}]_2 := \left[ \sum_{\ell \in [L] \setminus \{k\}} \mathbf{dk}_{\ell,k} \right]_2$ $[h_{2,k}]_2 := [\gamma_k]_2$ $\text{mpk} := (\mathcal{G}, [s]_1, [w]_1, [t]_2)$ $\text{hsk}_k := ([h_{1,k}]_2, [h_{2,k}]_2, \mathbf{F}_k)$ <b>return</b> $(\text{mpk}, (\text{hsk}_k)_{k \in [L]})$
<p><b>Dec</b>(<math>\text{sk}_k = [K]_2, \text{hsk}_k, \text{ct}</math>)</p> <hr/> $[D_0]_T := [C_1]_1 ([K]_2 + [h_1]_2)$ $[D_1]_T := [C_2]_1 [h_2]_2$ $[D_2]_T := \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} f_{i,j}^{(k)} \cdot ([C_{3,i}^\top]_1 [C_{4,j}]_2)$ <b>return</b> $[D_0]_T - [D_1]_T + [D_2]_T$	

**Fig. 2.** Weak RQFE construction.

<p><b>Setup</b>(<math>1^\lambda</math>)</p> <hr/> $\mathcal{G} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, [1]_1, [1]_2, \cdot) \leftarrow \text{GGen}(1^\lambda)$ <b>for</b> $\ell \in [L] : \mathbf{w}_\ell \leftarrow \mathbb{Z}_p^{n_1}; r_\ell \leftarrow \mathbb{Z}_p$ <b>return</b> $\text{crs} := \left( \mathcal{G}, ([\mathbf{w}_\ell]_1)_{\ell \in [L]}, ([r_\ell]_2)_{\ell \in [L]}, ([r_k \mathbf{w}_\ell]_2)_{k, \ell \in [L], k \neq \ell} \right)$	<p><b>KGen</b>(<math>\text{crs}, \ell</math>)</p> <hr/> $\text{sk}_\ell := \mathbf{v}_\ell \leftarrow \mathbb{Z}_p^n$ $\text{pk}_\ell := \left( [v_\ell]_1, ([r_k \mathbf{v}_\ell]_2)_{k \in [L] \setminus \{\ell\}} \right)$ <b>return</b> $(\text{pk}_\ell, \text{sk}_\ell)$
<p><b>Aggr</b>(<math>\text{crs}, (\text{pk}_\ell, \mathbf{y}_\ell)_{\ell \in [L]}</math>)</p> <hr/> <b>for</b> $k \in [L] :$ $[h_{0,k}]_2 := [r_k]_2$ $[h_{1,k}]_2 := \left[ r_k \sum_{\ell \in [L] \setminus \{k\}} (\mathbf{w}_\ell^\top + \mathbf{v}_\ell^\top) \mathbf{y}_\ell \right]_2$ $[\mathbf{h}_{2,k}^\top]_2 := \left[ r_k \sum_{\ell \in [L] \setminus \{k\}} \mathbf{w}_\ell^\top \right]_2$ $\text{mpk} := \left( \left[ \sum_{\ell \in [L]} (\mathbf{w}_\ell^\top + \mathbf{v}_\ell^\top) \mathbf{y}_\ell \right]_1, \left[ \sum_{\ell \in [L]} \mathbf{w}_\ell^\top \right]_1 \right)$ $\text{hsk}_k := ([h_{0,k}]_2, [h_{1,k}]_2, [\mathbf{h}_{2,k}^\top]_2)$ <b>return</b> $(\text{mpk}, (\text{hsk}_k)_{k \in [L]})$	<p><b>Enc</b>(<math>\text{mpk}, \mathbf{x} \in \mathbb{Z}_p^n</math>)</p> <hr/> $s \leftarrow \mathbb{Z}_p$ $[c_0]_1 := [s]_1$ $[c_1]_1 := \left[ s \sum_{\ell \in [L]} (\mathbf{w}_\ell^\top + \mathbf{v}_\ell^\top) \mathbf{y}_\ell \right]_1$ $[c_2^\top]_1 := \left[ s \sum_{\ell \in [L]} \mathbf{w}_\ell^\top + \mathbf{x}^\top \right]_1$ <b>return</b> $\text{ct} := ([c_0]_1, [c_1]_1, [c_2^\top]_1)$
<p><b>Dec</b>(<math>\text{sk}_k, \text{hsk}_k, \text{ct}</math>)</p> <hr/> $[d_0]_T := [c_1]_1 [h_0]_2 - [c_0]_1 [h_1]_2$ $[d_1]_T := [c_0]_1 [h_0]_2 \mathbf{v}_k^\top \mathbf{y}_k$ $[d_2^\top]_T := [c_2^\top]_1 [h_0]_2 - [c_0]_1 [\mathbf{h}_2^\top]_2$ <b>return</b> $[d_2^\top]_T \cdot \mathbf{y}_k - ([d_0]_T - [d_1]_T)$	

**Fig. 3.** RLFE construction.

**Theorem 1.** *RLFE (Fig. 3) is strongly compact (Definition 3).*

*Proof.* Assuming the groups description  $\mathcal{G}$  and each element in  $\mathbb{Z}_p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$  are of description size  $\text{poly}$ , we count the size of  $\text{mpk}$ ,  $\text{hsk}_\ell$ , and  $\text{ct}$ . We have  $|\text{mpk}|, |\text{hsk}_\ell|, |\text{ct}| = n \cdot \text{poly}$ . Notably, they are of size independent of  $L$ .

**Theorem 2.** *RLFE (Fig. 3) has relaxed correctness<sup>14</sup> (cf. Definition 2).*

*Proof.* Observe that for any decryptor  $k \in [L]$ ,

$$\begin{aligned} [d_0]_{\top} &= \left[ s \sum_{\ell \in [L]} (\mathbf{w}_\ell^{\top} + \mathbf{v}_\ell^{\top}) \mathbf{y}_\ell \right]_1 \left[ r_k \sum_{\ell \in [L] \setminus \{k\}} (\mathbf{w}_\ell^{\top} + \mathbf{v}_\ell^{\top}) \mathbf{y}_\ell \right]_2 \\ &= \left[ sr_k \sum_{\ell \in [L]} (\mathbf{w}_\ell^{\top} + \mathbf{v}_\ell^{\top}) \mathbf{y}_\ell \right]_{\top} - \left[ sr_k \sum_{\ell \in [L] \setminus \{k\}} (\mathbf{w}_\ell^{\top} + \mathbf{v}_\ell^{\top}) \mathbf{y}_\ell \right]_{\top} \\ &= [sr_k (\mathbf{w}_k^{\top} + \mathbf{v}_k^{\top}) \mathbf{y}_k], \end{aligned}$$

$$[d_1]_{\top} = [s]_1 [r_k]_2 \mathbf{v}_k^{\top} \mathbf{y}_k = [sr_k \mathbf{v}_k^{\top} \mathbf{y}_k],$$

$$\begin{aligned} [d_2^{\top}]_{\top} &= \left[ s \sum_{\ell \in [L]} \mathbf{w}_\ell^{\top} + \mathbf{x}^{\top} \right]_1 \left[ r_k \sum_{\ell \in [L] \setminus \{k\}} \mathbf{w}_\ell^{\top} \right]_2 \\ &= \left[ sr_k \sum_{\ell \in [L]} \mathbf{w}_\ell^{\top} + r_k \mathbf{x}^{\top} \right]_{\top} - \left[ sr_k \sum_{\ell \in [L] \setminus \{k\}} \mathbf{w}_\ell^{\top} \right]_{\top} = [sr_k \mathbf{w}_k^{\top} + r_k \mathbf{x}^{\top}]_{\top}. \end{aligned}$$

Therefore decryption outputs

$$\begin{aligned} & [d_2^{\top}]_{\top} \cdot \mathbf{y}_k - ([d_0]_{\top} - [d_1]_{\top}) \\ &= [sr_k \mathbf{w}_k^{\top} + r_k \mathbf{x}^{\top}]_{\top} \cdot \mathbf{y}_k - ([sr_k (\mathbf{w}_k^{\top} + \mathbf{v}_k^{\top}) \mathbf{y}_k]_{\top} - [sr_k \mathbf{v}_k^{\top} \mathbf{y}_k]_{\top}) \\ &= [sr_k \mathbf{w}_k^{\top} \mathbf{y}_k + r_k \mathbf{x}^{\top} \mathbf{y}_k]_{\top} - [sr_k \mathbf{w}_k^{\top} \mathbf{y}_k]_{\top} = [r_k \mathbf{x}^{\top} \mathbf{y}_k]_{\top}, \end{aligned}$$

as desired.

Our security proof relies on the following assumption.

**Assumption 3.** *Let  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, [1]_1, [1]_2) \leftarrow \text{GGen}(1^\lambda)$ . For any PPT  $\mathcal{A}$*

$$\left| \Pr \left[ \mathcal{A} \left( [s]_1, ([a_\ell]_1, [r_\ell]_2)_{\ell \in [L]}, ([r_k a_\ell]_2)_{k, \ell \in [L], k \neq \ell}, \left[ s \sum_{\ell \in [L]} a_\ell \right]_1 \right) = 1 \right] \right. \\ \left. - \Pr \left[ \mathcal{A} \left( [s]_1, ([a_\ell]_1, [r_\ell]_2)_{\ell \in [L]}, ([r_k a_\ell]_2)_{k, \ell \in [L], k \neq \ell}, [u]_1 \right) = 1 \right] \right|$$

*is negligible in  $\lambda$ , where  $s, u, a_\ell, r_\ell \leftarrow \mathbb{Z}_p$  for all  $\ell \in [L]$ .*

<sup>14</sup> Here, by relaxed, we mean that instead of  $[\mathbf{x}^{\top} \mathbf{y}_k]_{\top}$  the  $k$ -th decryptor would obtain  $[r_k \mathbf{x}^{\top} \mathbf{y}_k]_{\top}$ , which is perfectly bound to  $[\mathbf{x}^{\top} \mathbf{y}_k]_{\top}$  by the  $\text{crs}$ . We note that this relaxation does not affect the application of the RLFE to the bounded collusion RTT.

The above can be seen as a  $q$ -type variant (where  $q = L$ ) of the DDH assumption generalised into the bilinear group setting: Removing all elements in  $\mathbb{G}_2$ , the statement is implied by DDH (over  $\mathbb{G}_1$ ) which says that  $[s]_1 [a_\ell]_1 \approx_c \$$  for any  $\ell \in [L]$ .

*Remark 2.* In Assumption 3, it is important that  $\left[ s \sum_{\ell \in [L]} a_\ell \right]_1$  sums over all  $\ell \in [L]$  instead of any subset  $S \subset [L]$ . Otherwise, picking any  $k \notin S$ , one can distinguish  $\left[ s \sum_{\ell \in S} a_\ell \right]_1$  from random via the pairing equation  $\left[ s \sum_{\ell \in S} a_\ell \right]_1 [r_k]_2 \stackrel{?}{=} \sum_{\ell \in S} [s]_1 [r_k a_\ell]_2$ . With  $\left[ s \sum_{\ell \in [L]} a_\ell \right]_1$  instead, since  $[r_k a_k]_2$  for any  $k \in [L]$  is not given out, the same attack does not apply.

In the full version, we prove that Assumption 3 holds in the generic group model.

**Theorem 4.** *RLFE (Fig. 3) is selectively secure with static corruption (Definition 4) under Assumption 3.*

*Proof.* We define the following hybrids:

- $\mathcal{H}_{b,0}$ : This is same as the selective-security experiment for  $b \in \{0, 1\}$ , i.e. the distribution as in Fig. 3 encrypting  $\mathbf{x}_b^\top$ .
- $\mathcal{H}_{b,1}$ : Same as  $\mathcal{H}_{b,1}$ , except that we compute  $[\mathbf{c}_2^\top]_1$  as

$$\left[ t\mathbf{x}_1^\top + (1-t)\mathbf{x}_0^\top + s \sum_{\ell \in [L]} \mathbf{k}_\ell^\top \bar{\mathbf{Z}} \right]_1,$$

where  $(\mathbf{x}_0, \mathbf{x}_1)$  are the challenge messages from the adversary interacting with the experiment,  $t, s \in \mathbb{Z}_p$  and  $\mathbf{k}_\ell \in \mathbb{Z}_p^{n-1}$  are uniformly random, and  $\bar{\mathbf{Z}} \in \mathbb{Z}_p^{n-1 \times n}$  independent of  $b$  (defined below).

Notice that  $\mathcal{H}_{0,1} \equiv \mathcal{H}_{1,1}$ , since the distribution of all terms in  $[\mathbf{c}_2^\top]_1$ , hence also  $[\mathbf{c}_2^\top]_1$ , are independent of  $b$ . We show in the remaining that  $\mathcal{H}_{b,0} \approx_c \mathcal{H}_{b,1}$  under Assumption 3, which completes the proof.

Suppose there exists a PPT  $\mathcal{A}$  that distinguishes  $\mathcal{H}_{b,0}$  and  $\mathcal{H}_{b,1}$  with non-negligible probability. We construction a PPT  $\mathcal{B}$  against Assumption 3.

On input problem instance  $\left( [s]_1, ([a_\ell]_1, [r_\ell]_2)_{\ell \in [L]}, ([r_k a_\ell]_2)_{k, \ell \in [L], k \neq \ell}, [u]_1 \right)$  where  $[u]_1$  is either  $\left[ s \sum_{\ell \in [L]} a_\ell \right]_1$  or uniformly random,  $\mathcal{B}$  proceeds as follows:

- Receive the pair of challenge messages  $(\mathbf{x}_0, \mathbf{x}_1)$  and the set of corrupt users  $C \subseteq [L]$  from  $\mathcal{A}$ .
- Let  $\hat{\mathbf{x}} := \mathbf{x}_1 - \mathbf{x}_0$ , let  $\mathbf{B} := (\hat{\mathbf{x}} \mid \mathbf{Z})$  a basis of  $\mathbb{Z}_p^n$ , where  $\mathbf{Z}$  is arbitrary basis of the kernel space  $\hat{\mathbf{x}}^\perp$  of  $\hat{\mathbf{x}}$ .
- Sample random  $\mathbf{k}_\ell \leftarrow \mathbb{Z}_p^{n-1}$  for all  $\ell \in [L]$ .
- Pass crs to  $\mathcal{A}$  which is simulated as follows:

- For each  $\ell \in [L]$ , fetch  $[a_\ell]_1$  and  $([r_k]_2)_{k \in [L] \setminus \{\ell\}}$  from input and let

$$\begin{aligned} [\mathbf{w}_\ell]_1 &:= ([a_\ell]_1 \mid [\mathbf{k}_\ell^\top]_1) \mathbf{B}^{-1}, \\ [r_k \mathbf{w}_\ell^\top]_2 &:= ([r_k a_\ell]_2 \mid [r_k]_2 \mathbf{k}_\ell^\top) \mathbf{B}^{-1} \text{ for all } k \in [L] \setminus \{\ell\}. \end{aligned}$$

- Let  $\text{crs} := (\mathcal{G}, \{[\mathbf{w}_\ell]_1\}_{\ell \in [L]}, \{[r_\ell]_2\}_{\ell \in [L]}, \{[r_k \mathbf{w}_\ell]_2\}_{k, \ell \in [L], k \neq \ell})$ .
- For key query on user  $\ell \in [L]$ , if  $K[\ell] = \perp$ , same keys as follows:
  - If  $\ell \in [L] \setminus C$  is not corrupt: Sample random  $\mathbf{d}_\ell \leftarrow \mathbb{Z}_p^n$ , fetch  $[a_\ell]_1$  and  $([r_k]_2)_{k \in [L] \setminus \{\ell\}}$  from input, and compute  $\text{pk}_\ell := ([\mathbf{v}_\ell^\top]_1, [r_k \mathbf{v}_\ell^\top]_2)$  as

$$\begin{aligned} [\mathbf{v}_\ell^\top]_1 &:= [\mathbf{d}_\ell^\top]_1 - ([a_\ell \mid \mathbf{k}_\ell^\top]_1) \mathbf{B}^{-1}, \\ [r_k \mathbf{v}_\ell^\top]_2 &:= [r_k]_2 \mathbf{d}_\ell^\top - ([r_k a_\ell]_2 \mid [r_k]_2 \mathbf{k}_\ell^\top) \mathbf{B}^{-1}. \end{aligned}$$

- If  $\ell \in C$  is corrupt: Sample random  $\mathbf{v}_\ell \in \mathbb{Z}_p^n$ , fetch  $([r_k]_2)_{k \in [L] \setminus \{\ell\}}$  from input, let  $\text{pk}_\ell := ([\mathbf{v}_\ell^\top]_1, [r_k \mathbf{v}_\ell^\top]_2)$  and  $\text{sk}_\ell := \mathbf{v}_\ell$ .

Write the above to  $K[\ell]$  and answer accordingly.

- Receive the message  $(\mathbf{x}_0, \mathbf{x}_1)$ , the registrations  $(\text{pk}_\ell, \mathbf{y}_\ell)_{\ell \in [L]}$ , and randomness  $(\mathbf{v}_\ell)_{\ell \in M}$  for malicious parties from  $\mathcal{A}$ . Verify that (1)  $[L] \setminus M \subseteq K$ , (2) for each  $\ell \in M$  it holds that  $\text{pk}_\ell = ([\mathbf{v}_\ell^\top]_1, [r_k \mathbf{v}_\ell^\top]_2)$  for  $\mathbf{v}_\ell$  provided by  $\mathcal{A}$ , and (3) for all  $\ell \in C \cup M$  it holds that  $\mathbf{x}_0^\top \mathbf{y}_\ell = \mathbf{x}_1^\top \mathbf{y}_\ell$ . If all checks pass, let  $\text{sk}_\ell = \mathbf{v}_\ell$  for the malicious  $\ell \in M$ , and simulate the challenge ciphertext  $\text{ct}^*$  as follows:

- For each  $\ell \in C \cup M$ , write  $\mathbf{y}_\ell =: \mathbf{B} \begin{pmatrix} 0 \\ \tilde{\mathbf{y}}_\ell \end{pmatrix}$  where  $\tilde{\mathbf{y}}_\ell \in \mathbb{Z}_p^{n-1}$  (which is possible since  $\mathbf{x}_0^\top \mathbf{y}_\ell = \mathbf{x}_1^\top \mathbf{y}_\ell$ , equivalently  $\mathbf{y}_\ell \in \hat{\mathbf{x}}^\perp$ ).
- Fetch  $[s]_1$  and  $[u]_1$  from input, let  $\text{ct}^* := ([c_0]_1, [c_1]_1, [c_2]_1)$  where  $[c_0]_1 := [s]_1$  and

$$\begin{aligned} [c_1]_1 &:= [s]_1 \sum_{\ell \in [L] \setminus (C \cup M)} \mathbf{d}_\ell^\top \mathbf{y}_\ell + [s]_1 \sum_{\ell \in C \cup M} (\mathbf{k}_\ell^\top \tilde{\mathbf{y}}_\ell + \mathbf{v}_\ell^\top \mathbf{y}_\ell), \\ [c_2]_1 &:= \left( [u]_1 \mid [s]_1 \sum_{\ell \in [L]} \mathbf{k}_\ell^\top \right) \mathbf{B}^{-1} + [\mathbf{x}_b^\top]_1. \end{aligned}$$

- Pass  $\text{ct}^*$  to  $\mathcal{A}$  and return whatever  $\mathcal{A}$  returns.

We analyse the outputs of  $\mathcal{B}$ . First, notice that the simulated outputs can be expressed as setting

$$\begin{aligned} \mathbf{w}_\ell^\top &= (a_\ell \mid \mathbf{k}_\ell^\top) \mathbf{B}^{-1} \quad \text{for all } \ell \in [L] \\ \mathbf{v}_\ell^\top &= \mathbf{d}_\ell^\top - (a_\ell \mid \mathbf{k}_\ell^\top) \mathbf{B}^{-1} \quad \text{for all } \ell \in [L] \setminus (C \cup M), \end{aligned}$$

then computing all components except  $[c_2]_1$  in the same way as in the scheme. In more details, using the above two equations, the outputs can be expressed as

$$\begin{aligned} \text{crs} : \quad [r_k \mathbf{w}_\ell^\top]_2 &= [r_k (a_\ell \mid \mathbf{k}_\ell^\top) \mathbf{B}^{-1}]_2 = ([r_k a_\ell]_2 \mid [r_k]_2 \mathbf{k}_\ell^\top) \mathbf{B}^{-1} \\ \text{pk}_\ell, \ell \in [L] \setminus (C \cup M) : \quad [r_k \mathbf{v}_\ell^\top]_2 &= [r_k (\mathbf{d}_\ell^\top - (a_\ell \mid \mathbf{k}_\ell^\top) \mathbf{B}^{-1})]_2 \\ &= [r_k]_2 \mathbf{d}_\ell^\top - ([r_k a_\ell]_2 \mid [r_k]_2 \mathbf{k}_\ell^\top) \mathbf{B}^{-1} \end{aligned}$$

and for the challenge ciphertext  $\text{ct}^*$ , we have  $[c_1]_1$  being

$$\begin{aligned}
 & [s]_1 \sum_{\ell \in [L] \setminus (C \cup M)} \mathbf{d}_\ell^\top \mathbf{y}_\ell + [s]_1 \sum_{\ell \in (C \cup M)} (\mathbf{k}_\ell^\top \tilde{\mathbf{y}}_\ell + \mathbf{v}_\ell^\top \mathbf{y}_\ell) \\
 &= [s]_1 \sum_{\ell \in [L] \setminus (C \cup M)} (\mathbf{v}_\ell^\top + (a_\ell | \mathbf{k}_\ell^\top) \mathbf{B}^{-1}) \mathbf{y}_\ell + [s]_1 \sum_{\ell \in (C \cup M)} ((a_\ell | \mathbf{k}_\ell^\top) \mathbf{B}^{-1} \mathbf{y}_\ell + \mathbf{v}_\ell^\top \mathbf{y}_\ell) \\
 &= \left[ s \sum_{\ell \in [L]} ((a_\ell | \mathbf{k}_\ell^\top) \mathbf{B}^{-1} + \mathbf{v}_\ell^\top) \mathbf{y}_\ell \right]_1 = \left[ s \sum_{\ell \in [L]} (\mathbf{w}_\ell^\top + \mathbf{v}_\ell^\top) \mathbf{y}_\ell \right]_1
 \end{aligned}$$

where the second term in the second equality is by  $\mathbf{k}_\ell^\top \tilde{\mathbf{y}}_\ell = (a_\ell | \mathbf{k}_\ell^\top) \mathbf{B}^{-1} \mathbf{B} \begin{pmatrix} 0 \\ \tilde{\mathbf{y}}_\ell \end{pmatrix} = (a_\ell | \mathbf{k}_\ell^\top) \mathbf{B}^{-1} \mathbf{y}_\ell$ .

Since  $(a_\ell)_{\ell \in [L]}$ ,  $(\mathbf{k}_\ell)_{\ell \in [L]}$  and  $(\mathbf{d}_\ell)_{\ell \in [L] \setminus C}$  are all uniformly random, so are  $(\mathbf{w}_\ell)_{\ell \in [L]}$  and  $(\mathbf{v}_\ell)_{\ell \in [L] \setminus C}$ . The keys  $(\text{pk}_\ell, \text{sk}_\ell)$  for the corrupt users  $\ell \in C$  are computed honestly. Therefore, all components of the simulated  $\text{crs}$ ,  $\text{pk}_\ell, \text{sk}_\ell$  as well as  $[c_0]_1, [c_1]_1$  in  $\text{ct}^*$  are distributed same as in the scheme.

Finally we inspect  $[\mathbf{c}_2^\top]_1$ . Suppose  $[u]_1 = \left[ s \sum_{\ell \in [L]} a_\ell \right]_1$ , then

$$\begin{aligned}
 [\mathbf{c}_2^\top]_1 &= \left( \left[ s \sum_{\ell \in [L]} a_\ell \right]_1 \middle| \left[ s]_1 \sum_{\ell \in [L]} \mathbf{k}_\ell^\top \right) \mathbf{B}^{-1} + [\mathbf{x}_b^\top]_1 = \left[ s \sum_{\ell \in [L]} (a_\ell | \mathbf{k}_\ell^\top) \mathbf{B}^{-1} + \mathbf{x}_b^\top \right]_1 \\
 &= \left[ s \sum_{\ell \in [L]} \mathbf{w}_\ell^\top + \mathbf{x}_b^\top \right]_1
 \end{aligned}$$

which is exactly the ciphertext component encrypting  $\mathbf{x}_b$  as in the real scheme, or  $\mathcal{H}_{b,0}$ . Else if  $[u]_1$  is uniform, then write  $\mathbf{B}^{-1} =: \begin{pmatrix} \bar{\mathbf{x}}^\top \\ \bar{\mathbf{z}} \end{pmatrix}$ , and we have  $[\mathbf{c}_2^\top]_1$  being

$$\left( [u]_1 \middle| \left[ s]_1 \sum_{\ell \in [L]} \mathbf{k}_\ell^\top \right) \mathbf{B}^{-1} + [\mathbf{x}_b^\top]_1 = \left[ u \bar{\mathbf{x}}^\top + s \sum_{\ell \in [L]} \mathbf{k}_\ell^\top \bar{\mathbf{z}} + \mathbf{x}_0^\top + b(\mathbf{x}_1^\top - \mathbf{x}_0^\top) \right]_1.$$

Now observe  $\hat{\mathbf{x}}^\top = \hat{\mathbf{x}}^\top (\hat{\mathbf{x}} | \mathbf{z}) \begin{pmatrix} \bar{\mathbf{x}}^\top \\ \bar{\mathbf{z}} \end{pmatrix} = (\|\hat{\mathbf{x}}\|^2 | \hat{\mathbf{x}}^\top \mathbf{z}) \begin{pmatrix} \bar{\mathbf{x}}^\top \\ \bar{\mathbf{z}} \end{pmatrix} = \|\hat{\mathbf{x}}\|^2 \bar{\mathbf{x}}^\top + \underbrace{\hat{\mathbf{x}}^\top \mathbf{z} \bar{\mathbf{z}}}_{=0}$ , where  $\|\hat{\mathbf{x}}\|$  denotes the  $L_2$ -norm of  $\hat{\mathbf{x}}$ . Equivalently  $\bar{\mathbf{x}}^\top = c \hat{\mathbf{x}}^\top = c(\mathbf{x}_1^\top - \mathbf{x}_0^\top)$  where  $c := \|\hat{\mathbf{x}}\|^{-2}$ . Hence

$$\begin{aligned}
 [\mathbf{c}_2^\top]_1 &= \left[ uc(\mathbf{x}_1^\top - \mathbf{x}_0^\top) + \mathbf{x}_0^\top + b(\mathbf{x}_1^\top - \mathbf{x}_0^\top) + s \sum_{\ell \in [L]} \mathbf{k}_\ell^\top \bar{\mathbf{z}} \right]_1 \\
 &= \left[ t\mathbf{x}_1^\top + (1-t)\mathbf{x}_0^\top + s \sum_{\ell \in [L]} \mathbf{k}_\ell^\top \bar{\mathbf{z}} \right]_1,
 \end{aligned}$$

where  $t := uc + b$  is uniform over  $\mathbb{Z}_p$  since  $u$  is uniform and  $c \neq 0$  (since w.l.o.g.  $\hat{\mathbf{x}} \neq \mathbf{0}$ ). Therefore  $[\mathbf{c}_2^T]_1$  is distributed same as in  $\mathcal{H}_{b,1}$ .<sup>15</sup>

We conclude that  $\mathcal{B}$  perfectly simulates  $\mathcal{H}_{b,0}$  if the input  $[u]_1 = \left[ s \sum_{\ell \in [L]} a_\ell \right]_1$ , and perfectly simulates  $\mathcal{H}_{b,1}$  if  $[u]_1$  is uniformly random. The proof is completed.

## 5 Registered Traitor-Tracing

Traitor-tracing [10] is a cryptographic primitive that allows to identify users involved in illegal distribution of content. Below, we define and construct a registered version of traitor-tracing (RTT).

Our scheme is obtained by adapting existing transformations from quadratic functional encryption to traitor-tracing to the registered setting. We first show that the RQFE scheme of Sect. 4.2 implies predicate encryption for comparison (PEC) following [21]. The next step is just to recast PEC as a private linear broadcast encryption, a primitive first introduced in [7]. This in turn yields RTT by adapting the transformation presented in [7] to the registered setting.

In the full version, we have more results on RTT. Particularly, we show in how our RLFE scheme of Sect. 4.3 can be used to build an RTT scheme secure against bounded collusions. Further, we informally discuss revocation mechanisms for our RTT schemes.

### 5.1 Registered Private Linear Broadcast Encryption

We define and build a registered version of private linear broadcast encryption (PLBE), a primitive that was first defined in [7].

**Definition 5 (Registered Private Linear Broadcast Encryption).** *A registered private linear broadcast encryption (RPLBE) scheme for message space  $\mathcal{M}$ , ciphertext space  $\mathcal{C}$  and number of users  $L$  is a tuple of PPT algorithms (Setup, KGen, Aggr, Enc, TrEnc, Dec):*

- Setup( $1^\lambda$ ) *inputs the security parameter. It outputs a crs.*
- KGen(crs,  $\ell$ ) *inputs the crs and an index  $\ell \in [L]$ . It outputs a pair of public and secret keys  $(\mathbf{pk}_\ell, \mathbf{sk}_\ell)$  associated with the index  $\ell$ .*
- Aggr(crs,  $(\mathbf{pk}_\ell)_{\ell \in [L]}$ ) *inputs crs and public keys  $(\mathbf{pk}_\ell)_{\ell \in [L]}$ . It outputs a master public key  $\mathbf{mpk}$  and helper secret keys  $(\mathbf{hsk}_\ell)_{\ell \in [L]}$ .*
- Enc( $\mathbf{mpk}, m$ ) *inputs  $\mathbf{mpk}$  and a message  $m \in \mathcal{M}$ . It outputs a ciphertext  $\mathbf{ct} \in \mathcal{C}$ .*
- TrEnc( $\mathbf{mpk}, i, m$ ) *inputs  $\mathbf{mpk}$  an index  $i \in [L]$ , and a message  $m \in \mathcal{M}$ . It outputs a ciphertext  $\mathbf{ct} \in \mathcal{C}$ .*
- Dec( $\mathbf{sk}_\ell, \mathbf{hsk}_\ell, \mathbf{ct}$ ) *inputs a secret key  $\mathbf{sk}_\ell$ , a helper secret key  $\mathbf{hsk}_\ell$  and a ciphertext  $\mathbf{ct}$ . It outputs a message  $m'$ .*

<sup>15</sup> For any malicious user  $\ell \in M$ , decrypting  $[\mathbf{c}_2^T]_1$  in this case correctly yields  $\mathbf{x}_0^T \mathbf{y}_\ell = \mathbf{x}_1^T \mathbf{y}_\ell$  since  $(\mathbf{x}_1^T - \mathbf{x}_0^T) \mathbf{y}_\ell = \bar{\mathbf{Z}} \mathbf{y}_\ell = 0$  and  $\mathbf{c}_2^T \mathbf{y}_\ell = \left( \mathbf{x}_0^T + t(\mathbf{x}_1^T - \mathbf{x}_0^T) + s \sum_{\ell \in [L]} \mathbf{k}_\ell^T \bar{\mathbf{Z}} \right) \mathbf{y}_\ell = \mathbf{x}_0^T \mathbf{y}_\ell$ .



$\text{ExpRPLBE}_{\Pi, \mathcal{A}}^{x,b}(1^\lambda)$	$\text{KGenO}(\ell)$
$\text{crs} \leftarrow \text{Setup}(1^\lambda)$ $((\text{pk}_\ell)_{\ell \in [L]}, (\text{r}_\ell)_{\ell \in M}, i \in [L], (m_0, m_1)) \leftarrow \mathcal{A}^{\text{KGenO}(\cdot), \text{CorrO}(\cdot)}(\text{crs})$ assert $[L] \setminus M \subseteq K \subseteq [L]$ assert $\text{pk}_\ell \in \text{KGen}(\text{crs}, \ell; \text{r}_\ell) \quad \forall \ell \in M$ $(\text{mpk}, (\text{hsk}_\ell)_{\ell \in [L]}) \leftarrow \text{Aggr}(\text{crs}, (\text{pk}_\ell)_{\ell \in [L]})$ if $x = \text{Ind}$ : if $b = 0$ : $\text{ct}^* \leftarrow \text{Enc}(\text{mpk}, m_0)$ else : $\text{ct}^* \leftarrow \text{TrEnc}(\text{mpk}, 1, m_0)$ if $x = \text{MsgHide}$ : $\text{ct}^* \leftarrow \text{TrEnc}(\text{mpk}, L + 1, m_b)$ if $x = \text{IndexHide}$ : assert $\{i, i + 1\} \cap (C \cup M) = \emptyset$ $\text{ct}^* \leftarrow \text{TrEnc}(\text{mpk}, i + b, m_0)$ $b' \leftarrow \mathcal{A}(\text{ct}^*)$ return $b'$	if $K[\ell] = \perp$ $(\text{pk}_\ell, \text{sk}_\ell) \leftarrow \text{KGen}(\text{crs}, \ell)$ $K[\ell] := (\text{pk}_\ell, \text{sk}_\ell)$ $(\text{pk}_\ell, \text{sk}_\ell) \leftarrow K[\ell]$ return $\text{pk}_\ell$ <hr/> $\text{CorrO}(\ell)$ $C := C \cup \{\ell\}$ $(\text{pk}_\ell, \text{sk}_\ell) \leftarrow K[\ell]$ return $\text{sk}_\ell$

**Fig. 4.** Security experiments for RPLBE.

**Definition 6 (Correctness).** An RPLBE is correct if for all  $\lambda \in \mathbb{N}$ ,  $L \in \text{poly}$ ,  $m \in \mathcal{M}$ ,  $\text{crs} \in \text{Setup}(1^\lambda)$ ,  $(\text{pk}_\ell, \text{sk}_\ell) \in \text{KGen}(\text{crs}, \ell)$  where  $\ell \in [L]$ , and all  $i, j \in [L]$  such that  $i \leq j \leq L$ ,

$$\Pr \left[ m = m' \mid \begin{array}{l} (\text{mpk}, (\text{hsk}_\ell)_{\ell \in [L]}) \leftarrow \text{Aggr}(\text{crs}, (\text{pk}_\ell)_{\ell \in [L]}) \\ \text{ct} \leftarrow \text{TrEnc}(\text{mpk}, i, m) \\ m' \leftarrow \text{Dec}(\text{sk}_j, \text{hsk}_j, \text{ct}) \end{array} \right] = 1.$$

**Definition 7 (Indistinguishability, Message-Hiding, Index-Hiding).** An RPLBE scheme  $\Pi$  is said to be indistinguishable, message-hiding, and index-hiding respectively [7], if for all PPT  $\mathcal{A}$  it holds that

$$\left| \Pr \left[ \text{ExpRBLPE}_{\Pi, \mathcal{A}}^{x,0}(1^\lambda) = 1 \right] - \Pr \left[ \text{ExpRBLPE}_{\Pi, \mathcal{A}}^{x,1}(1^\lambda) = 1 \right] \right|$$

is negligible in  $\lambda$ , for  $x \in \{\text{Ind}, \text{MsgHide}, \text{IndexHide}\}$  respectively, where  $\text{Exp}_{\Pi, \mathcal{A}}^{x,b}$  is defined in Fig. 4.

To construct RPLBE, we first recall a lemma from [21] expressing the comparison predicate as a quadratic function.

**Lemma 1 ([21]).** Let  $L \in \text{poly}[\lambda]$ ,  $\ell \in [L]$ . Define the predicate  $F_\ell : [L + 1] \times \{1, 2\} \rightarrow \{0, 1, 2\}$ ,

$$F_\ell(i, m) = \begin{cases} m, & \text{if } i \leq \ell \\ 0, & \text{else} \end{cases}.$$

Then  $F_\ell(i, m) = \mathbf{x}_{i,m}^\top \mathbf{M}_\ell \mathbf{y}_{i,m}$  for some  $\mathbf{M}_\ell \in \{0, 1\}^{2\sqrt{L} \times (\sqrt{L}+1)}$  and

$$\mathbf{x}_{i,m} \in \{0, 1, 2\}^{2\sqrt{L}} \quad \text{and} \quad \mathbf{y}_{i,m} \in \{0, 1, 2\}^{\sqrt{L}+1}.$$

Moreover,  $\mathbf{M}_\ell$  is efficiently computable given  $\ell$ , and  $\mathbf{x}_{i,m}, \mathbf{y}_{i,m}$  are efficiently computable given  $(i, m)$ . The latter is denoted by  $(\mathbf{x}_{i,m}, \mathbf{y}_{i,m}) \leftarrow \mathcal{Z}(i, m)$ .

We sketch the proof and refer to [21] for a detailed analysis.

*Proof (Proof Sketch).* Let us assume that  $L \in \mathbb{N}$  is a perfect square for convenience and let  $\mathcal{Z}$  output  $(0^{2\sqrt{L}}, 0^{\sqrt{L}+1})$  if the input is  $(L+1, m)$  for any  $m$ . Clearly this yields  $F_\ell(L+1, m) = 0$  as wanted. In the rest we consider  $i \in [L]$ .

Fix any  $i \in [L]$ . Let  $(i_1, i_2) \in [\sqrt{L}] \times [\sqrt{L}]$  be such that  $i = (i_1 - 1)\sqrt{L} + i_2$  and define  $(\ell_1, \ell_2)$  analogously for  $\ell$ . Let

$$\tilde{\mathbf{v}} = (\mathbf{0}^{i_1}, \mathbf{1}^{\sqrt{L}-i_1}) \in \{0, 1\}^{\sqrt{L}} \quad \text{and} \quad \hat{\mathbf{v}} = \mathbf{e}_{i_1} \in \{0, 1\}^{\sqrt{L}}$$

where  $\mathbf{e}_{i_1}$  is the  $i_1$ -th unit vector. Furthermore, let

$$\bar{\mathbf{v}} = (\mathbf{0}^{i_2-1}, \mathbf{1}^{\sqrt{L}-i_2+1}) \in \{0, 1\}^{\sqrt{L}}.$$

For any  $j \in [\sqrt{L}]$ , denote  $\tilde{v}_j$  the  $j$ -th entry of  $\tilde{\mathbf{v}}$  and analogously for  $\hat{v}_j, \bar{v}_j$ . Now  $F_\ell(i, m) = m$  if and only if  $i \leq \ell$ , which implies either (1)  $i_1 < \ell_1$ , equivalently  $\tilde{v}_{\ell_1} = 1$ , or (2)  $i_1 = \ell_1$  and  $i_2 \leq \ell_2$ , equivalently  $\hat{v}_{\ell_1} \cdot \bar{v}_{\ell_2} = 1$ . That is,  $\tilde{v}_{\ell_1} + \hat{v}_{\ell_1} \bar{v}_{\ell_2} = 1$ . Thus, for any  $m \in \{1, 2\}$  and  $(\ell_1, \ell_2)$ , we can express  $m$  as  $m = m(\tilde{v}_{\ell_1} + \hat{v}_{\ell_1} \bar{v}_{\ell_2}) = \mathbf{x}_{i,m}^\top \mathbf{M}_\ell \mathbf{y}_{i,m}$ , for  $\mathbf{x}_{i,m}^\top := (m\tilde{\mathbf{v}}^\top, m\hat{\mathbf{v}}^\top) \in \{0, 1, 2\}^{2\sqrt{L}}$ ,  $\mathbf{y}_{i,m}^\top := (1, \bar{\mathbf{v}}^\top) \in \{0, 1, 2\}^{\sqrt{L}+1}$  and  $\mathbf{M}_\ell \in \{0, 1\}^{2\sqrt{L} \times (\sqrt{L}+1)}$  is as follows:

$$\mathbf{M}_\ell(r, c) = \begin{cases} 1, & \text{if } (r, c) = (\ell_1, 1) \text{ or } (r, c) = (\ell_1 + \sqrt{L}, \ell_2 + 1) \\ 0, & \text{else} \end{cases}.$$

We show that an RPLBE can be constructed using our weak RQFE in Fig. 2. Let  $L \in \text{poly}$  and  $\mathcal{M} = \{1, 2\}$ . For each  $\ell \in [L]$ , let function  $F_\ell$  and its corresponding matrix  $\mathbf{M}_\ell$  be as defined in Lemma 1. Also let  $\mathcal{Z}$  be as defined in Lemma 1. Let RQFE be the weak RQFE constructed in Fig. 2, with parameters  $n_1 = 2\sqrt{L}$ ,  $n_2 = \sqrt{L} + 1$ ,  $p > 2$ , and number of users  $L$ . In Fig. 5 we describe an RPLBE for the message space  $\mathcal{M}$  and  $L$  users.

Correctness of our construction follows directly from Lemma 1 and the correctness of RQFE. The next theorem states its security.

**Theorem 5 (Security).** *RPLBE (Fig. 5) is indistinguishable, message-hiding and index-hiding (Definition 7) if RQFE is secure.*

*Proof.* Indistinguishability follows trivially as both algorithms  $\text{TrEnc}(\text{mpk}, 1, m)$  and  $\text{Enc}(\text{mpk}, m)$  are exactly the same.

Message-hiding follows from the security of RQFE: By definition of  $F_{L+1}$  from Lemma 1, for any messages  $m_0, m_1 \in \mathcal{M}$  and  $\ell \in [L]$  we have  $F_\ell(L +$

<b>Setup</b> ( $1^\lambda$ )	<b>KGen</b> ( $\text{crs}, \ell$ )	<b>Aggr</b> ( $\text{crs}, \{\text{pk}_\ell\}_{\ell \in [L]}$ )
$\text{crs} \leftarrow \text{RQFE.Setup}(1^\lambda, \{\mathbf{M}_\ell\}_{\ell \in [L]})$ (Note: $\mathbf{F}_\ell$ in Fig. 2 is $\mathbf{M}_\ell$ here.)	$(\text{pk}_\ell, \text{sk}_\ell) \leftarrow \text{RQFE.KGen}(\text{crs}, \ell)$	$(\text{mpk}, \{\text{hsk}_\ell\}_{\ell \in L}) \leftarrow \text{RQFE.Aggr}(\text{crs}, \{\text{pk}_\ell\}_{\ell \in [L]})$
<b>return crs</b>	<b>return</b> $(\text{pk}_\ell, \text{sk}_\ell)$	<b>return</b> $(\text{mpk}, \{\text{hsk}_\ell\}_{\ell \in L})$
<b>Enc</b> ( $\text{mpk}, m \in \{1, 2\}$ )	<b>TrEnc</b> ( $\text{mpk}, i, m \in \{1, 2\}$ )	<b>Dec</b> ( $\text{sk}_\ell, \text{hsk}_\ell, \text{ct}$ )
$(\mathbf{x}_{1,m}, \mathbf{y}_{1,m}) \leftarrow \mathcal{Z}(1, m)$	$(\mathbf{x}_{i,m}, \mathbf{y}_{i,m}) \leftarrow \mathcal{Z}(i, m)$	$[m]_{\top} \leftarrow \text{RQFE.Dec}(\text{sk}_\ell, \text{hsk}_\ell, \text{ct})$
$\text{ct} \leftarrow \text{RQFE.Enc}(\text{mpk}, (\mathbf{x}_{1,m}, \mathbf{y}_{1,m}))$	$\text{ct} \leftarrow \text{RQFE.Enc}(\text{mpk}, (\mathbf{x}_{i,m}, \mathbf{y}_{i,m}))$	if $[1]_{\top} = [m]_{\top}$ : <b>return</b> 1
<b>return ct</b>	<b>return ct</b>	else : <b>return</b> 2

**Fig. 5.** RPLBE construction.

$1, m_0) = F_\ell(L + 1, m_1) = 0$ , hence by security of RQFE, the adversary learns nothing more than 0 in either experiment.

Index-hiding also follows from the security of RQFE: The index  $i$  or  $i + 1$  is encoded only in the RQFE message as  $(\mathbf{x}_{i,m}, \mathbf{y}_{i,m})$  or  $(\mathbf{x}_{i+1,m}, \mathbf{y}_{i+1,m})$ . For any index  $\ell \in C \cup M$  of which the adversary has the secret key, it holds that  $i \neq \ell$ , therefore either  $i < i + 1 \leq \ell$  so that  $F_\ell(i, m) = F_\ell(i + 1, m) = m$ , or  $i + 1 > i > \ell$  so that  $F_\ell(i) = F_\ell(i + 1) = 0$ . Thus by security of RQFE, a ciphertext encrypting  $(\mathbf{x}_{i,m}, \mathbf{y}_{i,m})$  is indistinguishable from one encrypting  $(\mathbf{x}_{i+1,m}, \mathbf{y}_{i+1,m})$ .

*Message Space for RPLBE:* Recall the proof of Lemma 1 at a high level. A registered user for any slot  $\ell$  in our RPLBE (Fig. 5) computes a comparison predicate  $P_\ell$  (outputting a bit). Note that the message  $m \in \{1, 2\}$  is embedded in the function evaluation as  $m \cdot P_\ell(\mathbf{x})$ . If  $\mathcal{M} = \{0, 1\}$ , we cannot distinguish the two cases: i)  $P_\ell(\mathbf{x}) = 0$  with  $m = 1$  and ii)  $P_\ell(\mathbf{x}) = 1$  with  $m = 0$ . Hence, we set  $\mathcal{M} = \{1, 2\}$  in order to distinguish the above cases if the predicate  $P_\ell$  is satisfied or not.

*Supporting Large Message Spaces:* As explained above, our RPLBE decryption (Fig. 5) for any slot  $\ell$  yields  $m \cdot P_\ell(\mathbf{x})$  for a message  $m \in \{1, 2\}$  and a comparison predicate  $P_\ell$ . The message space is set to  $\mathcal{M} = \{1, 2\}$  for conceptual simplicity. However, our scheme can easily support larger messages with a standard transformation: The encryptor samples  $m \leftarrow \mathbb{Z}_q$  and uses  $[m]$  as a one-time symmetric key to encrypt an arbitrarily long message. Decryptor recovers  $[m]$  if  $P_\ell(\mathbf{x}) = 1$ . The only overhead is that of a symmetric encryption scheme (e.g., AES), which is very fast.

*Optimizations.* In practice, a short, random seed  $\in \{0, 1\}^\lambda$  can be used as the  $\text{crs}$  along with a pseudorandom generator with a sufficient stretch, which gives a transparent setup for the RQFE.

Further, our RPLBE requires RQFE to compute quadratic functions associated to highly sparse, binary matrices. Lemma 1 explicitly characterises this:  $\forall k \in [L]$ ,  $\mathbf{M}_k$  contains exactly two 1s at positions  $(k_1, 1)$  and  $(k_1 + \sqrt{L}, k_2 + 1)$  for the natural map  $k \mapsto (k_1, k_2)$  as specified above. We show how this significantly reduces the number of operations in the KGen and Dec algorithms:

The KGen algorithm (Fig. 2) for each user  $\ell \in [L]$  can compute the terms

$$[\mathbf{dk}_{\ell,k}]_2 = s_{\ell,k_1} [t_1]_2 + s_{\ell,k_1+\sqrt{L}} [t_{k_2+1}]_2 + w_\ell [\gamma_k]_2$$

for matrix  $\mathbf{M}_k$  with randomness

$$(\mathbf{s}_\ell, w_\ell) \in \mathbb{Z}_p^{2\sqrt{L}+1},$$

where  $s_{\ell,i}, [t_j]_2$  denote the  $i$ -th and  $j$ -th elements in  $\mathbf{s}_\ell$  and  $[t]_2$  respectively. This reduces computing *each* cross-term to only a constant number of operations (precisely, 3 exponentiations and 2 group operations in  $\mathbb{G}_2$ ). Similarly, the slot  $k$  decryptor (Fig. 2) can avoid computing the full pairing-product in the term  $[D_2]_{\mathbb{T}}$ . Instead, it can simply compute it as

$$\left( [\mathbf{C}_{3,k_1}^{\mathbb{T}}]_1 [\mathbf{C}_{4,1}]_2 \right) + \left( [\mathbf{C}_{3,k_1+\sqrt{L}}^{\mathbb{T}}]_1 [\mathbf{C}_{4,k_2+1}]_2 \right).$$

So the decryptor also need not parse the full ciphertext (that grows with  $\sqrt{L}$ ). Rather, it needs to parse only 10 group elements, namely:

$$[\mathbf{C}_1]_1, [\mathbf{C}_2]_1, \left( [\mathbf{C}_{3,k_1}]_1, [\mathbf{C}_{3,k_1+\sqrt{L}}]_1 \right), \left( [\mathbf{C}_{4,1}]_2, [\mathbf{C}_{4,k_2+1}]_2 \right)$$

Computing  $[D_2]_{\mathbb{T}}$  requires just 4 pairings reducing its total count to only 6 during decryption (along with 5 group operations in  $\mathbb{G}_{\mathbb{T}}$  and 1 in  $\mathbb{G}_2$ ). Crucially, the total number of operations is *independent* of all  $\sqrt{L}$  factors and is a constant.

Further, note that an index  $i \in [L]$  is encoded during encryption using *binary* vectors  $(\mathbf{x}_{i,m}, \mathbf{y}_{i,m})$  (where  $\mathbf{x}_{i,m}$  is also scaled with the message  $m \in \{1, 2\}$ ). Hence, one can further optimise the number of operations in the Enc, TrEnc algorithms based on  $i$  and its equivalently encoded vectors  $\tilde{\mathbf{v}}, \hat{\mathbf{v}}$  and  $\bar{\mathbf{v}}$  as shown in Lemma 1.

## 5.2 Registered Traitor-Tracing

We are now ready to define and build registered traitor-tracing. The definitions and construction from RPLBE largely follow the one from [7], except that we now work in the registered setting. We provide the definitions, construction and proofs below.

**Definition 8 (Registered Traitor-Tracing).** *A registered traitor-tracing (RTT) scheme for a message space  $\mathcal{M}$ , ciphertext space  $\mathcal{C}$  and number of users  $L$  consists of the tuple of PPT algorithms (Setup, KGen, Aggr, Enc, Trace<sup>D</sup>, Dec):*

- Setup( $1^\lambda$ ) inputs the security parameter. It outputs a crs.
- KGen(crs,  $\ell$ ) inputs the crs and an index  $\ell \in [L]$ . It outputs a pair of public and secret keys  $(\mathbf{pk}_\ell, \mathbf{sk}_\ell)$  associated with the index  $\ell$ .
- Aggr(crs,  $(\mathbf{pk}_\ell)_{\ell \in [L]}$ ) inputs the crs and public keys  $(\mathbf{pk}_\ell)_{\ell \in [L]}$ . It outputs a master public key  $\mathbf{mpk}$  and helper secret keys  $(\mathbf{hsk}_\ell)_{\ell \in [L]}$ .
- Enc( $\mathbf{mpk}, m$ ) inputs  $\mathbf{mpk}$  and a message  $m \in \mathcal{M}$ . It outputs a ciphertext  $\mathbf{ct} \in \mathcal{C}$ .

- $\text{Trace}^D(\text{mpk}, \epsilon)$  inputs  $\text{mpk}$  and a parameter  $\epsilon$ . It has oracle access to a decoder  $D$ . It outputs an identity  $i \in [L]$ .
- $\text{Dec}(\text{sk}_\ell, \text{hsk}_\ell, \text{ct})$  inputs a secret key  $\text{sk}_\ell$ , a helper secret key  $\text{hsk}_\ell$  and a ciphertext  $\text{ct}$ . It outputs a message  $m'$ .

**Definition 9 (Correctness).** An RTT is said to be correct if for all  $\lambda \in \mathbb{N}$ ,  $L \in \text{poly}$ ,  $m \in \mathcal{M}$ ,  $k \in [L]$ ,  $\text{crs} \in \text{Setup}(1^\lambda)$ ,  $(\text{pk}_\ell, \text{sk}_\ell) \in \text{KGen}(\text{crs}, \ell)$  where  $\ell \in [L]$ , it holds that

$$\Pr \left[ m = m' \mid \begin{array}{l} (\text{mpk}, (\text{hsk}_\ell)_{\ell \in [L]}) \leftarrow \text{Aggr}(\text{crs}, (\text{pk}_\ell)_{\ell \in [L]}) \\ \text{ct} \leftarrow \text{Enc}(\text{mpk}, m) \\ m' \leftarrow \text{Dec}(\text{sk}_k, \text{hsk}_k, \text{ct}) \end{array} \right] = 1.$$

**Definition 10 (Semantic Security and Traceability).** An RTT is said to be semantically secure, if for any PPT  $\mathcal{A}$  it holds that

$$|\Pr [\text{ExpRTT-Security}_{\Pi, \mathcal{A}}^0(1^\lambda) = 1] - \Pr [\text{ExpRTT-Security}_{\Pi, \mathcal{A}}^1(1^\lambda) = 1]|$$

is negligible in  $\lambda$ , and traceable against arbitrary collusion, if for any PPT  $\mathcal{A}$

$$\Pr [\text{ExpRTT-Traceability}_{\Pi, \mathcal{A}}(1^\lambda) = 1] \leq \text{negl}(\lambda),$$

where  $\text{ExpRTT-Security}_{\Pi, \mathcal{A}}^b$  and  $\text{ExpRTT-Traceability}_{\Pi, \mathcal{A}}$  are defined in Fig. 6.

We also consider a selective security with static corruption version of the  $\text{ExpRTT-Security}_{\Pi, \mathcal{A}}^b(1^\lambda)$  experiment, where the adversary  $\mathcal{A}$  announces the messages  $(m_0, m_1)$  and the corruption set at the beginning of the experiment, i.e. before seeing  $\text{crs}$ . Similarly, a scheme is said to be traceable with static corruption if the adversary in  $\text{ExpRTT-Traceability}_{\Pi, \mathcal{A}}$  announces the corruption set at the beginning of the experiment.

In Fig. 7 we present an RTT scheme based on an RPLBE scheme RPLBE, which is similar to that in [7] but recast in the registered setting. Its correctness follows directly from that of RPLBE.

**Theorem 6 (Semantic security).** RTT (Fig. 7) is semantically secure (Definition 10) if RPLBE is indistinguishable, message-hiding and index-hiding.

*Proof.* The proof follows the same reasoning as the one from [7].

*Hybrid  $\mathcal{H}_0$ .* In this hybrid, the challenger sets  $b = 0$ .

*Hybrid  $\mathcal{H}_1$ .* This hybrid is identical to the previous one except that we set  $\text{ct} \leftarrow \text{RPLBE.TrEnc}(\text{mpk}, 1, m_0)$ . Indistinguishability of hybrids follow from the indistinguishability of RPLBE.

*Hybrid  $\mathcal{H}_2$ .* This hybrid is identical to the previous one except that we set  $\text{ct} \leftarrow \text{RPLBE.TrEnc}(\text{mpk}, L+1, m_0)$ . This is done via a sequence of sub-hybrids, where we replace  $\text{ct} \leftarrow \text{RPLBE.TrEnc}(\text{mpk}, i, m_0)$  by  $\text{ct} \leftarrow \text{RPLBE.TrEnc}(\text{mpk}, i+1, m_0)$ , for all  $i \in [L]$ . Indistinguishability follow from the RPLBE index-hiding security.

ExpRTT-Security $_{\Pi, \mathcal{A}}^b(1^\lambda)$	ExpRTT-Traceability $_{\Pi, \mathcal{A}}(1^\lambda)$
$\text{crs} \leftarrow \text{Setup}(1^\lambda)$ $(m_0, m_1) \leftarrow \mathcal{A}(\text{crs})$ $(\text{pk}_\ell, \text{sk}_\ell) \leftarrow \text{KGen}(\text{crs}, \ell) \forall \ell \in [L]$ $(\text{mpk}, (\text{hsk}_\ell)_{\ell \in [L]}) \leftarrow \text{Aggr}(\text{crs}, (\text{pk}_\ell)_{\ell \in [L]})$ $\text{ct}^* \leftarrow \text{Enc}(\text{mpk}, m_b)$ <b>return</b> $\mathcal{A}(\text{ct}^*)$	$\text{crs} \leftarrow \text{Setup}(1^\lambda)$ $((\text{pk}_\ell)_{\ell \in [L]}, (\text{r}_\ell)_{\ell \in M}, \text{D}) \leftarrow \mathcal{A}^{\text{KGenO}(\cdot), \text{CorrO}(\cdot)}(\text{crs})$ <b>assert</b> $[L] \setminus M \subseteq K \subseteq [L]$ <b>assert</b> $\text{pk}_\ell \in \text{KGen}(\text{crs}, \ell; \text{r}_\ell) \forall \ell \in M$ $(\text{mpk}, (\text{hsk}_\ell)_{\ell \in [L]}) \leftarrow \text{Aggr}(\text{crs}, (\text{pk}_\ell)_{\ell \in [L]})$ $S^* \leftarrow \text{Trace}^{\text{D}}(\text{mpk}, \epsilon)$ $b_1 := (\text{Pr}[m \leftarrow \text{D}(\text{Enc}(\text{mpk}, m)) \mid m \leftarrow_{\$} \mathcal{M}] > \epsilon)$ $b_2 := (S^* = \emptyset \vee S^* \not\subseteq C)$ <b>return</b> $(b_1 \wedge b_2)$
$\text{KGenO}(\ell)$ <hr style="width: 100%;"/> <b>if</b> $K[\ell] = \perp$ $(\text{pk}_\ell, \text{sk}_\ell) \leftarrow \text{KGen}(\text{crs}, \ell)$ $K[\ell] := (\text{pk}_\ell, \text{sk}_\ell)$ $(\text{pk}_\ell, \text{sk}_\ell) \leftarrow K[\ell]$ <b>return</b> $\text{pk}_\ell$	$\text{CorrO}(\ell)$ <hr style="width: 100%;"/> $C := C \cup \{\ell\}; (\text{pk}_\ell, \text{sk}_\ell) \leftarrow K[\ell]$ <b>return</b> $\text{sk}_\ell$

**Fig. 6.** Security experiments for RTT.

Trace $^{\text{D}}(\text{mpk}, \epsilon)$
$W := 8\lambda(L/\epsilon)^2$ <b>for</b> $i \in [L + 1]$ : $\text{count} := 0$ <b>for</b> $w \in [W]$ : $m \leftarrow_{\$} \{0, 1\}$ $\text{ct} \leftarrow \text{RPLBE.TrEnc}(\text{mpk}, i, m)$ <b>if</b> $\text{D}(\text{ct}) = m$ : $\text{count} := \text{count} + 1$ $\hat{p}_i := \text{count}/W$ $S := \{i \in [L] : \hat{p}_i - \hat{p}_{i+1} \geq \epsilon/(4L)\}$ <b>return</b> $i \leftarrow_{\$} S$

**Fig. 7.** Trace $^{\text{D}}$  algorithm of the RTT construction from RPLBE. Other algorithms (Setup, KGen, Enc, Dec, Aggr) of the RTT are identical to those of RPLBE.

*Hybrid  $\mathcal{H}_3$ .* This hybrid is identical to the previous one except that we set  $\text{ct} \leftarrow \text{RPLBE.TrEnc}(\text{mpk}, L + 1, m_1)$ . Indistinguishability of hybrids follow from the message-hiding of RPLBE.

*Hybrid  $\mathcal{H}_4$ .* This hybrid is identical to the previous one except that we set  $\text{ct} \leftarrow \text{RPLBE.TrEnc}(\text{mpk}, 1, m_1)$ . Indistinguishability of hybrids follow from the index-hiding of RPLBE.

*Hybrid  $\mathcal{H}_5$ .* This hybrid is identical to the previous one except that we set  $\text{ct} \leftarrow \text{RPLBE.Enc}(\text{mpk}, m_1)$ . Indistinguishability of hybrids follow from the indistinguishability of RPLBE.

**Theorem 7 (Traceability).** *RTT (Fig. 7) is traceable against arbitrary collusion (Definition 10) if RPLBE is indistinguishable, message-hiding and index-hiding.*

*Proof.* The proof follows the same reasoning as the one from [7]. We sketch the main ideas here and refer to [7] for a more detailed analysis (it is straightforward to adapt their proof to the registered setting).

Let  $\epsilon > 0$  be a constant. Denote by  $p_i = \Pr[\text{D}(\text{RPLBE.TrEnc}(\text{mpk}, i, m)) = m]$  and  $p = \Pr[\text{D}(\text{RPLBE.Enc}(\text{mpk}, m)) = m]$ . The proof is divided into 3 different types of adversaries.

- Type 1: D is a  $\epsilon$ -useful decoder for which  $|p - p_1| > 1/P(\lambda)$  for some polynomial  $P$ .
- Type 2: D is a  $\epsilon$ -useful decoder for which  $|p - p_1| \leq \text{negl}(\lambda)$  but Trace outputs an empty set.
- Type 3: D is a  $\epsilon$ -useful decoder for which  $|p - p_1| \leq \text{negl}(\lambda)$  but Trace outputs a set which is not contained in the set of colluders.

An adversary of type 1 can be used to break indistinguishability of the underlying RPLBE. An adversary of type 2 can be used to break message-hiding of the underlying RPLBE. Finally, an adversary of type 3 can be used to break the index-hiding of the underlying RPLBE.

*Efficiency.* Instantiating Fig. 7 with the RPLBE in Fig. 5 via our weak RQFE (Fig. 2), we obtain a concretely efficient RTT scheme. Recall that the functions  $\mathbf{F}_\ell$  used for RTT can be succinctly described from Lemma 1. Moreover, the  $\text{crs}$  consists of random elements that can be succinctly described by a short seed to be expanded using a random oracle.

## 6 Benchmarks

We implemented a prototype<sup>16</sup> of our RPLBE scheme (Sect. 5.1) in Python. As explained in Sect. 5, RPLBE immediately implies a registered traitor-tracing, without any modification to the algorithms. For the implementation we set  $L$  to be a perfect square, and we ran our benchmarks with different values of  $L \in \{16, 64, 256, 1024\}$ . We calculated the time it took to run the Setup and Aggr for each  $L$ . For the KGen and Enc and Dec we calculated the average times for each slot, over 100 repetitions of the experiment. The benchmarks were conducted on a personal computer with a AMD Ryzen 5 5600X 3.7 GHz CPU and 32 GB of RAM running Arch Linux with kernel 6.7.1-arch1-1. In Table 2 we report the measurements for our benchmarks plotted in Fig. 8.

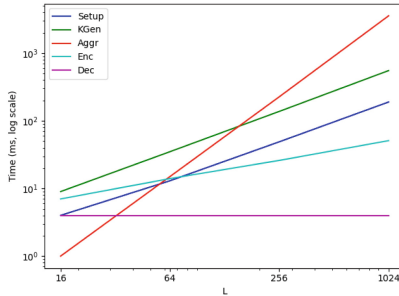
<sup>16</sup> <https://github.com/ahmadrezarahimi/RPLBE>.

*Storage.* The storage requirement of our RPLBE is quite modest: In the RPLBE scheme with  $L = 1024$ , we calculated the sizes of the expanded  $\text{crs}$ ,  $\text{mpk}$ , and the ciphertext, and they were 135 KB, 6.6 KB and 6.7 KB respectively. Furthermore, the sizes of a user’s public key, secret key, and helper secret key are 102.5 KB, 97B, and 194B, respectively.

*Group Operations.* For the choice of pairings, we used the BLS12-381 elliptic curve via the `petrelc` [32] Python wrapper around RELIC [4]: each element in  $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$  is represented with 49, 97, and 384 bytes, respectively. On our machine, exponentiation in  $\mathbb{G}_1$  and  $\mathbb{G}_2$  takes an average of 6.6 and 5.8  $\mu\text{s}$ , respectively, and each pairing evaluation takes 0.64 ms.

**Table 2.** Runtimes of our RPLBE algorithms for different  $L$ .

$L$	Time (ms)				
	Setup	KGen	Aggr	Enc	Dec
16	3.86	9.04	1.06	7.26	4.04
64	13.31	35.14	14.56	13.53	4.04
256	48.94	138.17	226.93	26.11	4.04
1024	189.57	553.87	3576.37	51.24	4.04



**Fig. 8.** Runtime plots of RPLBE algorithms with a growing number of users, interpolated from the measurements taken from  $L = \{16, 64, 256, 1024\}$ . Both axes are in log-scale.

## References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography. Lecture Notes in Computer Science, vol. 9020, pp. 733–751. Springer, Heidelberg, Germany, Gaithersburg, MD, USA (Mar 30 – Apr 1, 2015)



2. Agrawal, S., Bhattacharjee, S., Phan, D.H., Stehlé, D., Yamada, S.: Efficient public trace and revoke from standard assumptions: Extended abstract. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017: 24th Conference on Computer and Communications Security. pp. 2277–2293. ACM Press, Dallas, TX, USA (Oct 31 – Nov 2, 2017)
3. Agrawal, S., Kumari, S., Yadav, A., Yamada, S.: Broadcast, trace and revoke with optimal parameters from polynomial hardness. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023. pp. 605–636. Springer Nature Switzerland, Cham (2023)
4. Aranha, D.F., Gouvêa, C.P.L., Markmann, T., Wahby, R.S., Liao, K.: RELIC is an Efficient Library for Cryptography. <https://github.com/relic-toolkit/relic> (2020)
5. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017, Part I. Lecture Notes in Computer Science, vol. 10401, pp. 67–98. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2017)
6. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) Advances in Cryptology – CRYPTO 2001. Lecture Notes in Computer Science, vol. 2139, pp. 1–18. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001)
7. Boneh, D., Sahai, A., Waters, B.: Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay, S. (ed.) Advances in Cryptology – EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 573–592. Springer, Heidelberg, Germany, St. Petersburg, Russia (May 28 – Jun 1, 2006)
8. Boneh, D., Waters, B.: A fully collusion resistant broadcast, trace, and revoke system. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006: 13th Conference on Computer and Communications Security. pp. 211–220. ACM Press, Alexandria, Virginia, USA (Oct 30 – Nov 3, 2006)
9. Boneh, D., Zhandry, M.: Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 480–499. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2014)
10. Chor, B., Fiat, A., Naor, M.: Tracing traitors. In: Desmedt, Y. (ed.) Advances in Cryptology – CRYPTO’94. Lecture Notes in Computer Science, vol. 839, pp. 257–270. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 21–25, 1994)
11. Cong, K., Eldefrawy, K., Smart, N.P.: Optimizing registration based encryption. In: Paterson, M.B. (ed.) Cryptography and Coding. pp. 129–157. Springer International Publishing, Cham (2021)
12. Datta, P., Pal, T.: Registration-based functional encryption. IACR Cryptol. ePrint Arch. p. 457 (2023), <https://eprint.iacr.org/2023/457>
13. Datta, P., Pal, T., Yamada, S.: Registered fe beyond predicates: (attribute-based) linear functions and more. Cryptology ePrint Archive, Paper 2023/457 (2023), <https://eprint.iacr.org/2023/457>, <https://eprint.iacr.org/2023/457>
14. Dodis, Y., Fazio, N.: Public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In: Desmedt, Y. (ed.) PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography. Lecture Notes in Computer Science, vol. 2567, pp. 100–115. Springer, Heidelberg, Germany, Miami, FL, USA (Jan 6–8, 2003)

15. Döttling, N., Kolonelos, D., Lai, R.W.F., Lin, C., Malavolta, G., Rahimi, A.: Efficient laconic cryptography from  $\mathbb{A}$  learning with  $\mathbb{A}$  errors. In: Hazay, C., Stam, M. (eds.) *Advances in Cryptology – EUROCRYPT 2023*. pp. 417–446. Springer Nature Switzerland, Cham (2023)
16. Fiore, D., Kolonelos, D., de Perthuis, P.: Cuckoo commitments: Registration-based encryption and key-value map commitments for large spaces. In: *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security*, Beijing, China, December 4–8, 2023, Proceedings. Lecture Notes in Computer Science, Springer (2023)
17. Francati, D., Friolo, D., Maitra, M., Malavolta, G., Rahimi, A., Venturi, D.: Registered (inner-product) functional encryption. In: *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security*, Beijing, China, December 4–8, 2023, Proceedings. Lecture Notes in Computer Science, Springer (2023)
18. Freitag, C., Waters, B., Wu, D.J.: How to  $\mathbb{A}$  use (plain) witness encryption: Registered abe, flexible broadcast, and more. In: Handschuh, H., Lysyanskaya, A. (eds.) *Advances in Cryptology – CRYPTO 2023*. pp. 498–531. Springer Nature Switzerland, Cham (2023)
19. Garg, S., Hajiabadi, M., Mahmoody, M., Rahimi, A.: Registration-based encryption: Removing private-key generator from IBE. In: Beimel, A., Dziembowski, S. (eds.) *TCC 2018: 16th Theory of Cryptography Conference, Part I*. Lecture Notes in Computer Science, vol. 11239, pp. 689–718. Springer, Heidelberg, Germany, Panaji, India (Nov 11–14, 2018)
20. Garg, S., Hajiabadi, M., Mahmoody, M., Rahimi, A., Sekar, S.: Registration-based encryption from standard assumptions. In: Lin, D., Sako, K. (eds.) *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*. Lecture Notes in Computer Science, vol. 11443, pp. 63–93. Springer, Heidelberg, Germany, Beijing, China (Apr 14–17, 2019)
21. Gay, R.: Functional encryption for quadratic functions, and applications to predicate encryption. *Cryptology ePrint Archive*, Report 2016/1106 (2016), <http://eprint.iacr.org/2016/1106>
22. Glaeser, N., Kolonelos, D., Malavolta, G., Rahimi, A.: Efficient registration-based encryption. *Cryptology ePrint Archive*, Paper 2022/1505 (2022), <https://eprint.iacr.org/2022/1505>
23. Gong, J., Luo, J., Wee, H.: Traitor tracing with  $N^{1/3}$ -size ciphertexts and  $O(1)$ -size keys from  $k$ -Lin. In: Hazay, C., Stam, M. (eds.) *Advances in Cryptology – EUROCRYPT 2023*. pp. 637–668. Springer Nature Switzerland, Cham (2023)
24. Goyal, R., Koppula, V., Waters, B.: Collusion resistant traitor tracing from learning with errors. In: Diakonikolas, I., Kempe, D., Henzinger, M. (eds.) *50th Annual ACM Symposium on Theory of Computing*. pp. 660–670. ACM Press, Los Angeles, CA, USA (Jun 25–29, 2018)
25. Goyal, R., Koppula, V., Waters, B.: New approaches to traitor tracing with embedded identities. In: Hofheinz, D., Rosen, A. (eds.) *TCC 2019: 17th Theory of Cryptography Conference, Part II*. Lecture Notes in Computer Science, vol. 11892, pp. 149–179. Springer, Heidelberg, Germany, Nuremberg, Germany (Dec 1–5, 2019)
26. Goyal, R., Vusirikala, S.: Verifiable registration-based encryption. In: Shacham, H., Boldyreva, A. (eds.) *Advances in Cryptology – CRYPTO 2020, Part I*. pp. 621–651. Lecture Notes in Computer Science, Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2020)





27. Hohenberger, S., Lu, G., Waters, B., Wu, D.J.: Registered attribute-based encryption. In: Hazay, C., Stam, M. (eds.) *Advances in Cryptology – EUROCRYPT 2023*. pp. 511–542. Springer Nature Switzerland, Cham (2023)
28. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. pp. 60–73 (2021)
29. Kim, C.H., Hwang, Y.H., Lee, P.J.: An efficient public key trace and revoke scheme secure against adaptive chosen ciphertext attack. In: Lai, C.S. (ed.) *Advances in Cryptology – ASIACRYPT 2003*. *Lecture Notes in Computer Science*, vol. 2894, pp. 359–373. Springer, Heidelberg, Germany, Taipei, Taiwan (Nov 30 – Dec 4, 2003)
30. Kim, S., Wu, D.J.: Collusion resistant trace-and-revoke for arbitrary identities from standard assumptions. In: *Advances in Cryptology – ASIACRYPT 2020, Part II*. pp. 66–97. *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany (Dec 2020)
31. Kolonelos, D., Malavolta, G., Wee, H.: Distributed broadcast encryption from bilinear groups. *Cryptology ePrint Archive*, Paper 2023/874 (2023), <https://eprint.iacr.org/2023/874>, <https://eprint.iacr.org/2023/874>
32. Laurent Girod, W.L.: petrelic is a python wrapper around relic. <https://github.com/spring-epfl/petrelic> (2022)
33. Luo, J.: Ad hoc (decentralized) broadcast, trace, and revoke. *Cryptology ePrint Archive*, Paper 2022/925 (2022), <https://eprint.iacr.org/2022/925>, <https://eprint.iacr.org/2022/925>
34. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) *Advances in Cryptology – CRYPTO 2001*. *Lecture Notes in Computer Science*, vol. 2139, pp. 41–62. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001)
35. Naor, M., Pinkas, B.: Efficient trace and revoke schemes. In: Frankel, Y. (ed.) *FC 2000: 4th International Conference on Financial Cryptography*. *Lecture Notes in Computer Science*, vol. 1962, pp. 1–20. Springer, Heidelberg, Germany, Anguilla, British West Indies (Feb 20–24, 2001)
36. Nishimaki, R., Wichs, D., Zhandry, M.: Anonymous traitor tracing: How to embed arbitrary information in a key. In: Fischlin, M., Coron, J.S. (eds.) *Advances in Cryptology – EUROCRYPT 2016, Part II*. *Lecture Notes in Computer Science*, vol. 9666, pp. 388–419. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016)
37. Rogaway, P.: The moral character of cryptographic work. *Cryptology ePrint Archive*, Report 2015/1162 (2015), <http://eprint.iacr.org/2015/1162>
38. Wahby, R.S., Boneh, D.: Fast and simple constant-time hashing to the bls12-381 elliptic curve. *IACR Transactions on Cryptographic Hardware and Embedded Systems 2019(4)*, 154–179 (Aug 2019), <https://tches.iacr.org/index.php/TCHES/article/view/8348>
39. Wee, H.: Functional encryption for quadratic functions from  $k$ -lin, revisited. In: *TCC 2020: 18th Theory of Cryptography Conference, Part I*. pp. 210–228. *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany (Mar 2020)
40. Wu, Q., Qin, B., Zhang, L., Domingo-Ferrer, J.: Ad hoc broadcast encryption. In: *Proceedings of the 17th ACM Conference on Computer and Communications Security*. p. 741–743. *CCS '10*, Association for Computing Machinery, New York, NY, USA (2010), <https://doi.org/10.1145/1866307.1866416>

41. Zhandry, M.: New techniques for traitor tracing: Size  $N^{1/3}$  and more from pairings. In: Shacham, H., Boldyreva, A. (eds.) *Advances in Cryptology – CRYPTO 2020, Part I*. pp. 652–682. *Lecture Notes in Computer Science*, Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2020)
42. Zhu, Z., Li, J., Zhang, K., Gong, J., Qian, H.: Registered functional encryptions from pairings. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. pp. 373–402. Springer (2024)
43. Zhu, Z., Zhang, K., Gong, J., Qian, H.: Registered abe via predicate encodings. In: *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 4-8, 2023, Proceedings. Lecture Notes in Computer Science*, Springer (2023)

# **Threshold Cryptography**



# Interactive Threshold Mercurial Signatures and Applications

Masayuki Abe<sup>1,2</sup>, Masaya Nanri<sup>2</sup>, Octavio Perez Kempner<sup>1</sup>,  
and Mehdi Tibouchi<sup>1,2</sup>

<sup>1</sup> NTT Social Informatics Laboratories, Tokyo, Japan  
{msyk.abe, octavio.perezkempner, mehdi.tibouchi}@ntt.com

<sup>2</sup> Kyoto University, Kyoto, Japan  
masaya.nanri@icloud.com

**Abstract.** Mercurial signatures are an extension of equivalence class signatures that allow malleability for the public keys, messages, and signatures within the respective classes. Unfortunately, the most efficient construction to date suffers from a weak public key class-hiding property, where the original signer with the signing key can link the public keys in the same class. This is a severe limitation in their applications, where the signer is often considered untrustworthy of privacy.

This paper presents two-party and multi-party *interactive threshold mercurial signatures* that overcome the above limitation by eliminating the single entity who knows the signing key. For the general case, we propose two constructions. The first follows the same interactive structure as the two-party case, avoiding complex distributed computations such as randomness generation, inversion, and multiplication, and even eliminates the need for private communication between parties. The second is based on a blueprint for general multi-party computation using verifiable secret sharing, but adopting optimizations.

We show applications in anonymous credential systems that individually fit the two-party and multi-party constructions. In particular, in the two-party case, our approach provides stronger privacy by completely removing the trust in the authorities. We also discuss more applications, from blind signatures to multi-signatures and threshold ring signatures.

Finally, to showcase the practicality of our approach, we implement our interactive constructions and compare them against related alternatives.

**Keywords:** Mercurial Signatures · Equivalence Class Signatures · Threshold Signatures · Class-Hiding · Anonymous Credentials

## 1 Introduction

Equivalence Class Signatures (EQS) [43, 45] are malleable signatures [23] defined over a vector of group elements. They are structure-preserving [3, 4] and, thus, equipped with a bilinear pairing so that public keys and signatures also consist of group elements. This allows them to be verified evaluating pairing-product equations without requiring any specific encoding. They have been extensively used as a building block for many cryptographic primitives, including anonymous

credentials (*e.g.*, [28, 35, 43, 45, 46]), blind signatures [41, 42], group signatures [8, 9, 36] and sanitizable signatures [21] to name a few. Related primitives include signatures with flexible public keys [8] and Mercurial Signatures (MS) [28, 32, 33, 50], which is the main focus of this paper.

EQS allow one to randomize a signature, adapting it to a new message and a new public key in the same equivalence class. Security requires adapted signatures to look like freshly computed ones (signature adaption) and some notion of unlinkability when adapting messages and public keys (also referred to as class-hiding). In many applications, the adversary does not know the discrete logarithms of the message vector, and thus, *message class-hiding* is implied by the decisional Diffie-Hellman assumption. Recently, Bauer and Fuchsbaauer [10] proposed an EQS construction based on the idea of signatures on randomizable ciphertexts [14], achieving a stronger notion of message class-hiding covering the case in which the adversary knows the discrete logarithms of the message vector. Consequently, for message class-hiding, all possible scenarios are well-studied.

Regarding public key class-hiding, however, no satisfactory solution has been put forth so far. All known constructions only provide public key class-hiding as long as the adversary does not know the signing key. In other words, the original signer must be trusted. This is most evident for anonymous credentials where MS have been used to provide issuer-hiding features [27, 28, 50] and to build delegatable schemes [32, 33] where the issuer has to be trusted for issuer-hiding. That is, given a valid key pair  $(sk, pk)$  and a randomized public key  $pk'$  of  $pk$ , the issuer with  $sk$  can determine whether  $pk'$  is related to  $pk$ . Thus, issuers can identify if a credential has been issued to a user belonging to their organization, even if they do not know specifically to whom. While this can be tolerated in some scenarios, it can suffice to fully de-anonymize users in others. The situation is even worse for delegatable credentials because every user in the credential chain must be trusted. Otherwise, an adversary can identify chains containing a corrupted user by recognizing randomized keys.

## 1.1 Our Contributions

We propose *Threshold Mercurial Signature* (TMS) schemes where signing keys are distributed among signers, and a quorum cooperates to produce mercurial signatures. With secure distributed key generation, no signer below the threshold knows the key. This ensures no sub-threshold parties can access the secret key, justifying the weak public key class-hiding property and broadening the privacy-preserving applications of MS. Our contributions are summarized as follows.

1. **Two-party Mercurial Signature Scheme (Sect. 4):** We develop a distributed two-party signing protocol for the MS scheme described in [32, 43]. Two signers with additively shared signing keys interact with each other to generate a mercurial signature on a given message. The protocol consists of three sequenced moves and is secure against static corruption. This minimal setting is not only essential for illustrating our ideas for eliminating expensive

distributed computations but also serves a crucial role in issuer-hiding anonymous credentials, eliminating the need for trustworthy issuers (see Sect. 1.2 and 4.4 for more discussion and details). This solves an open problem of anonymous credentials based on EQS [28].

2. **Multi-party Mercurial Signature Scheme (Sect. 5):** We generalize the two-party protocol to the  $t$ -out-of- $n$  threshold setting. It is not a straightforward task due to the asymmetric nature of our two-party protocol. In this protocol, the three moves of interaction in the two-party case are simulated with the signers lined up in order, taking input from the previous signer and sending the result of local computation to the next one. A malicious signer in the sequence complicates security analysis, but the essential idea remains unchanged. As an application, we demonstrate how this  $t$ -out-of- $n$  scheme is useful for delegatable credentials (see Sect. 5.3 for details)<sup>1</sup>.
3. **Experimental Evaluation (Sect. 6):** The actual efficiency depends on the instantiation of underlying zero-knowledge proofs of knowledge and optimization of the group operations. Besides, the computational complexity scales linearly with the number of parties. For this reason, we implement our protocols and report benchmarks considering different numbers of parties and application settings. The overhead is relatively minor compared to the implementation of the original MS, allowing us to produce signatures in less than 0.5 s for practical scenarios involving ten parties.

## 1.2 Technical Overview

*Distributed Signing:* In the MS from [32], defined over pairing groups generated by  $G$  and  $\hat{G}$ , a signature  $(Z, Y, \hat{Y})$  on message  $M$  is computed as  $Z = M^{xy}$ ,  $Y = G^{1/y}$ ,  $\hat{Y} = \hat{G}^{1/y}$  with signing key  $x$  and ephemeral randomness  $y$ . Let  $[x]$  denote additive (or polynomial) shares of  $x$ , and consider the signers having  $[x]$  collaborate to compute a signature. A naive approach would be to generate shared randomness  $[y]$ , compute shared product  $[xy]$  and shared inverse  $[1/y]$ , and reconstruct  $xy$  and  $1/y$  on the exponent of  $M$ ,  $G$  and  $\hat{G}$ . This would require three invocations of distributed key generation (DKG) protocols involving commitments or verifiable secret sharing to avoid rushing adversaries that attempt to bias the resulting signature. Since this can be a cumbersome task for both two-party and multi-party cases, we first consider constructions that do not require such machinery. Instead, we observe that the bias caused by a rushing adversary can be ignored since mercurial signatures are malleable. In brief, the recipient can remove the bias by adapting the signature. In light of this observation, our first proposal incorporates the following techniques:

- We generate ephemeral randomness  $y$  in a multiplicative manner. Denote the multiplicative sharing by  $\langle y \rangle$ . This makes shared inversion  $\langle 1/y \rangle$  a local computation. Computing  $M^{xy}$  could be done first by computing  $M^x$  using  $[x]$  and then compute  $(M^x)^y$  using  $\langle y \rangle$  in sequence.

---

<sup>1</sup> More applications are discussed in the full version ([5], Appendix B).



- However, the above method leaks intermediate value  $M^x$  that prevents the security proof from going through. We develop efficient blind computation of  $M^{xy}$  where  $M^x$  is blinded by random factor  $Y^r$  and unblinded with  $G^r$ . Namely, the signers first compute  $Y^r M^x$  and then  $(Y^r M^x)^y$ , which can be done efficiently by the sequence of local computations. Since  $(Y^r M^x)^y = G^r M^{xy}$ , unblinding it with  $G^r$  results in  $M^{xy}$  as desired.
- In the threshold case, where more than two parties are involved in the signing process, we found that the randomness  $r$  mentioned above is insufficient to simulate more than two honest parties simultaneously. To address this issue, we introduce additional randomness into the signing protocol without altering its fundamental structure. This is achieved by incorporating random additive shares of zero into the intermediate computations, which cancel out when  $Y^r M^x$  is computed correctly. We develop a technique to generate these shares solely through public communication between the parties.

We also consider several optimizations to the naive approach based on multi-party computation and propose a second construction based on Abe’s multiplication protocol [1], which nicely fits our needs as explained in Sect. 5.2.

*Enhancing Issuer-Hiding in Anonymous Credentials:* The authorities’ role in a credential system with MS is to issue a signature on the user’s attributes. As discussed earlier, however, authorities are trusted for privacy in the sense that they do not abuse their signing key to trace the signatures. Plug-in replacement of MS with our TMS immediately raises the bar for violating users’ privacy. Nevertheless, threshold authorities are assumed to not collude to retain privacy.

We eliminate such an unverifiable trust using our TMS. In our issuing protocol, the authority and user Alice engage in the two-party TMS. The resulting signature verifies with the joint public key from the authority and Alice. When Alice anonymously shows the credential, she proves in zero knowledge that the randomized joint key properly includes a valid, authoritative public key, and she knows the randomized secret key for the remainder. This way, Alice can protect her privacy by herself without trusting the authority.

### 1.3 Related Work

**Mercurial Signatures.** There are two constructions of MS in the literature: one by Crites and Lysyanskaya [32] and another by Connolly *et al.* [28]. The MS from [28] was recently shown to be flawed in [11, 12], and it is broken. In Sect. 2.2, we recall the construction from [32]. As mentioned, it presents a major drawback, as any signer can track randomizations of previously issued signatures. This is because a public key  $pk$  is a vector of elements, and any randomization is just a multiplication in the exponent by the same randomization factor  $\rho$ . Hence, given knowledge of a secret key  $sk$  and any  $pk'$ , it suffices to multiply  $pk'$  in the exponent by the inverse of  $sk$ . Consequently, if all elements are the same, it must be the case that  $pk'$  is a randomization of  $pk$  for some  $\rho$ . Our work presents a threshold version for [32] that, instead of getting a multiplicative share in the

exponent of each element in the public key, we get an additive share. As a result, we can provide a stronger class-hiding notion, as further discussed in Sect. 4.4.

**Pointcheval-Sanders Signatures.** Very recently, Sanders and Traoré [59] proposed a modified version of Pointcheval-Sanders (PS) signatures [56, 57] to build an efficient issuer-hiding mechanism for anonymous credentials with strong security guarantees. Their approach consists of letting credential verifiers define an access policy for a set of issuers. More precisely, users take the verifier’s access policy to adapt their signature to verify if and only if the policy is satisfied (*i.e.*, the user’s signature/credential was signed by one of the issuers in the set). For security, verifiers must compute a zero-knowledge proof attesting to the correct computation of their access policy for the issuers’ set. In other words, this approach can be seen as letting each verifier define a custom common reference string (CRS) as their access policy, and the zero-knowledge proof attests to the correct computation of said CRS. Our approach to anonymous credentials resembles [59], and we borrow their NIKZ proof. However, in our case, verifiers only specify the issuer’s set as their access policy, and our solution does not require any proof of knowledge for the hidden attributes during the showing. Furthermore, we provide backward compatibility with previous attribute-based credentials constructions from EQS that provide revocation and auditability features [27, 35], potentially covering a more comprehensive range of functionalities.

**Threshold Signatures.** The ongoing NIST standardization effort related to threshold signatures [19] motivated many recent works tackling different settings, *e.g.*, [13, 29–31, 60]. Considering pairing-based constructions, threshold versions of the BLS signature [17, 18] such as [16] have gained significant attention over the past years, with security proven in the adaptive setting [7, 13, 34]. Threshold versions of BLS can be verified as a regular signature and are key-randomizable [37]. However, they are not structure-preserving and cannot be used as an alternative for EQS/MS. This is the first work to address the construction of threshold schemes for EQS. Closely related work to ours by Crites *et al.* [29] presented (non-interactive) Threshold Structure-Preserving Signatures. Their motivation was to have a drop-in replacement for standard SPS in the threshold setting. While the non-interactive setting is attractive and allows the authors to propose constructions compatible with the UC framework, this comes at the cost of using an indexed message space. In particular, a relatively new assumption called *Indexed Diffie-Hellman Message Space* is required to prove security. Very recent work by Mitrokotsa *et al.* [51] overcomes the previous limitation of [29] by removing the need of an indexed space. However, we stress that none of these works are EQS (let alone MS). Moreover, the constructions provided are not even randomizable. The indexed message space used in [29] defines an equivalence class, but this does not carry over to the TSPS construction (for a message  $m$ ,  $\hat{G}^m$  always stays as is, and thus,  $m$  is fixed). Looking at [51], it is a tag-based construction whose tag is not randomizable (see  $\sigma_1$  in [51]).

We take a different approach considering an interactive signing process. As we show, our approach offers several advantages for different applications where non-interactive TSPS fall short. Thus, our contribution broadens the scope of threshold SPS to include EQS, opening new research directions (see, for instance, the case of threshold ring signatures [20] from TMS discussed in the full version.

**Multi-signatures.** Multi-signatures are a special case of threshold signatures where the threshold  $t = n$ . Recent work mostly focuses on pairing-free and non-interactive constructions (e.g., [6, 13, 38, 53]), compatible with existing deployments in the blockchain sphere. Our approach is more general and focused on privacy-preserving applications that could benefit from malleable signatures with added functionalities and stronger security properties.

## 2 Preliminaries

**Notation.** The set of integers  $1, 2, \dots, n$  is denoted  $[n]$ . We call  $\mathbb{Z}_p$  the ring of integers modulus  $p$  if  $p \in \mathbb{N}$ . For a set  $\mathcal{S}$  and  $r \in \mathcal{S}$ ,  $r \leftarrow_{\mathcal{S}}$  denotes that  $r$  has been sampled uniformly randomly from  $\mathcal{S}$ . The security parameter  $\kappa$  is usually passed in unary form. We use  $\lambda$  for Lagrange coefficients, and we denote the adversary's state by  $\text{st}$ . Let  $\text{BGen}$  be a PPT algorithm that on input  $1^\kappa$ , returns public parameters  $\text{pp} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, G, \hat{G}, e)$  describing an asymmetric bilinear group where  $\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T$  are cyclic groups of prime order  $p$  with  $\lceil \log_2 p \rceil = \kappa$ ,  $G$  and  $\hat{G}$  are generators of  $\mathbb{G}$  and  $\hat{\mathbb{G}}$ , and  $e : \mathbb{G} \times \hat{\mathbb{G}} \rightarrow \mathbb{G}_T$  is an efficiently computable (non-degenerate) bilinear map.  $\text{pp}$  is considered Type-III if no efficiently computable isomorphism between  $\mathbb{G}$  and  $\hat{\mathbb{G}}$  is known. We then assume that the following DDH assumption in  $\mathbb{G}$  holds for  $\text{BGen}$ , as well as  $\hat{\mathbb{G}}$ .

**DDH Assumption.** Let  $\text{BGen}$  be a bilinear group generator that outputs public parameters  $\text{pp}$ . The *decisional Diffie-Hellman assumption* holds relative to  $\mathbb{G}$  for  $\text{BGen}$ , if for all p.p.t adversaries  $\mathcal{A}$  the following probability is negligible,

$$\Pr \left[ \begin{array}{l} \text{pp} \leftarrow_{\mathcal{S}} \text{BGen}(1^\kappa); r, s, t \leftarrow_{\mathcal{S}} \mathbb{Z}_p; b \leftarrow_{\mathcal{S}} \{0, 1\} \\ b^* \leftarrow_{\mathcal{S}} \mathcal{A}(\text{pp}, G^r, G^s, G^{(1-b)t+brs}) \end{array} : b^* = b \right] - \frac{1}{2}$$

### 2.1 Zero-Knowledge Proofs of Knowledge

We require secure Zero-Knowledge Proofs of Knowledge (ZKPoK) that are complete, zero-knowledge, and knowledge sound. Many instantiations are available in different models and with different assumptions, directly affecting our protocols' security. In this paper, for presentation and performance, we consider a non-interactive form of zero-knowledge proofs that allows online witness extraction. It is available in the random oracle model for a stand-alone execution where our implementation resorts. Alternatively, the ZKPoK's can be instantiated via interactive five-round PoK's in the standard model [44]. We leave the evaluation of other variants, such as using Fischlin's transform [26, 40], for future research.

## 2.2 Mercurial Signatures

We recap syntax and security notions of MS as presented in [32]. We recall that MS are EQS that support key randomization. Let  $\mathcal{R}$  be an equivalence relation where  $[x]_{\mathcal{R}} = \{y \mid \mathcal{R}(x, y)\}$  denotes the equivalence class of which  $x$  is a representative. As in [32], we will loosely consider parametrized relations and say they are well-defined as long as the corresponding parameters are.

**Definition 1 (Mercurial signature).** A MS scheme for parametrized equivalence relations  $\mathcal{R}_M, \mathcal{R}_{pk}, \mathcal{R}_{sk}$  is a tuple of the following polynomial-time algorithms, which are deterministic algorithms unless otherwise stated:

$\text{PGen}(1^\kappa) \rightarrow \text{pp}$ : On input the security parameter  $1^\kappa$ , this PPT algorithm outputs the public parameters  $\text{pp}$ . This includes parameters for the parametrized equivalence relations  $\mathcal{R}_M, \mathcal{R}_{pk}$ , and  $\mathcal{R}_{sk}$  so they are well-defined. It also includes parameters for the algorithms  $\text{sample}_\rho$  and  $\text{sample}_\mu$ , which sample key and message converters, respectively.

$\text{KGen}(\text{pp}, \ell) \rightarrow (\text{pk}, \text{sk})$ : On input the public parameters  $\text{pp}$  and a length parameter  $\ell$ , this PPT algorithm outputs a key pair  $(\text{pk}, \text{sk})$ . The message space  $\mathcal{M}$  is well-defined from  $\text{pp}$  and  $\ell$ . This algorithm also defines a correspondence between public and secret keys: we write  $(\text{pk}, \text{sk}) \in \text{KGen}(\text{pp}, \ell)$  if there exists a set of random choices that  $\text{KGen}$  could make to output  $(\text{pk}, \text{sk})$ .

$\text{Sign}(\text{pp}, \text{sk}, M) \rightarrow \sigma$ : On input the signing key  $\text{sk}$  and a message  $M \in \mathcal{M}$ , this PPT algorithm outputs a signature  $\sigma$ .

$\text{Verify}(\text{pp}, M, \sigma, \text{pk}) \rightarrow 0/1$ : On input the public key  $\text{pk}$ , a message  $M \in \mathcal{M}$ , and a purported signature  $\sigma$ , output 0 or 1.

$\text{ConvertSK}(\text{sk}, \rho) \rightarrow \text{sk}'$ : On input  $\text{sk}$  and a key converter  $\rho \in \text{sample}_\rho$ , output a new secret key  $\text{sk}' \in [\text{sk}]_{\mathcal{R}_{sk}}$ .

$\text{ConvertPK}(\text{pk}, \rho) \rightarrow \text{pk}'$ : On input  $\text{pk}$  and a key converter  $\rho \in \text{sample}_\rho$ , output a new public key  $\text{pk}' \in [\text{pk}]_{\mathcal{R}_{pk}}$ .

$\text{ConvertSig}(\text{pk}, M, \sigma, \rho) \rightarrow \sigma'$ : On input  $\text{pk}$ , a message  $M \in \mathcal{M}$ , a signature  $\sigma$ , and key converter  $\rho \in \text{sample}_\rho$ , this PPT algorithm returns a new signature  $\sigma'$ .

$\text{ChgRep}(\text{pk}, M, \sigma, \mu) \rightarrow (M', \sigma')$ : On input  $\text{pk}$ , a message  $M \in \mathcal{M}$ , a signature  $\sigma$ , and a message converter  $\mu \in \text{sample}_\mu$ , this PPT algorithm computes a new message  $M' \in [M]_{\mathcal{R}_M}$  and a new signature  $\sigma'$  and outputs  $(M', \sigma')$ .

**Definition 2 (Correctness).** A MS scheme for parametrized equivalence relations  $\mathcal{R}_M, \mathcal{R}_{pk}, \mathcal{R}_{sk}$  is correct if it satisfies the following conditions for all  $\kappa$ , for all  $\text{pp} \in \text{PGen}(1^\kappa)$ , for all  $\ell > 1$ , for all  $(\text{pk}, \text{sk}) \in \text{KGen}(\text{pp}, \ell)$ :

- Verification:  $\forall M \in \mathcal{M}, \forall \sigma \in \text{Sign}(\text{sk}, M), \text{Verify}(\text{pk}, M, \sigma) = 1$ .
- Key conversion:  $\forall \rho \in \text{sample}_\rho, (\text{ConvertPK}(\text{pk}, \rho), \text{ConvertSK}(\text{sk}, \rho)) \in \text{KGen}(\text{pp}, \ell)$ . Moreover,  $\text{ConvertSK}(\text{sk}, \rho) \in [\text{sk}]_{\mathcal{R}_{sk}}$  and  $\text{ConvertPK}(\text{pk}, \rho) \in [\text{pk}]_{\mathcal{R}_{pk}}$ .
- Signature conversion:  $\forall M \in \mathcal{M}, \forall \sigma$  such that  $\text{Verify}(\text{pk}, M, \sigma) = 1, \forall \rho \in \text{sample}_\rho, \forall \sigma' \in \text{ConvertSig}(\text{pk}, M, \sigma, \rho), \text{Verify}(\text{ConvertPK}(\text{pk}, \rho), M, \sigma') = 1$ .
- Change of message representative:  $\forall M \in \mathcal{M}, \forall \sigma$  such that  $\text{Verify}(\text{pk}, M, \sigma) = 1, \forall \mu \in \text{sample}_\mu, \text{Verify}(\text{pk}, M', \sigma') = 1$ , where  $(M', \sigma') = \text{ChgRep}(\text{pk}, M, \sigma, \mu)$ . Moreover,  $M' \in [M]_{\mathcal{R}_M}$ .

**Definition 3 (Unforgeability).** A MS scheme for parametrized equivalence relations  $\mathcal{R}_M, \mathcal{R}_{pk}, \mathcal{R}_{sk}$  is unforgeable if for all polynomial-length parameters  $\ell(\kappa)$  and any PPT adversary  $\mathcal{A}$  having access to a signing oracle, the following probability is negligible,

$$Pr \left[ \begin{array}{l} pp \leftarrow_s \text{PGen}(1^\kappa) \\ (sk, pk) \leftarrow_s \text{KGen}(pp, \ell(\kappa)) \\ (pk^*, M^*, \sigma^*) \leftarrow \mathcal{A}^{\text{Sign}(sk, \cdot)}(pk) \end{array} : \begin{array}{l} \forall M \in Q, [M^*]_{\mathcal{R}_M} \neq [M]_{\mathcal{R}_M} \\ \wedge [pk^*]_{\mathcal{R}_{pk}} = [pk]_{\mathcal{R}_{pk}} \\ \wedge \text{Verify}(M^*, \sigma^*, pk^*) = 1 \end{array} \right],$$

where  $Q$  is the set of queries that  $\mathcal{A}$  has issued to the signing oracle.

**Definition 4 (Class-Hiding).** A MS scheme is class-hiding if it satisfies the following two properties:

- *Message class-hiding:* if the advantage of any PPT adversary  $\mathcal{A}$  defined by  $\text{Adv}_{\text{MS}, \mathcal{A}}^{\text{MSG-CH}}(\kappa) := 2 \cdot Pr[\text{Exp}_{\text{MS}, \mathcal{A}}^{\text{MSG-CH}}(\kappa) \Rightarrow \text{true}] - 1 = \epsilon(\kappa)$ .
- *Public key class-hiding:* if the advantage of any PPT adversary  $\mathcal{A}$  defined by  $\text{Adv}_{\text{MS}, \mathcal{A}}^{\text{PK-CH}}(\kappa) := 2 \cdot Pr[\text{Exp}_{\text{MS}, \mathcal{A}}^{\text{PK-CH}}(\kappa) \Rightarrow \text{true}] - 1 = \epsilon(\kappa)$ .

The experiments  $\text{Exp}_{\text{MS}, \mathcal{A}}^{\text{MSG-CH}}(\kappa)$  and  $\text{Exp}_{\text{MS}, \mathcal{A}}^{\text{PK-CH}}(\kappa)$  are defined as follows:

Experiment $\text{Exp}_{\text{MS}, \mathcal{A}}^{\text{MSG-CH}}(\kappa)$	
$pp \leftarrow_s \text{PGen}(1^\kappa); b \leftarrow_s \{0, 1\}; M_1 \leftarrow_s \mathcal{M}; M_2^0 \leftarrow_s \mathcal{M}; M_2^1 \leftarrow_s [M_1]_{\mathcal{R}_M}$	
$b' \leftarrow_s \mathcal{A}(pp, M_1, m_2^b); \text{return } b = b'$	
Experiment $\text{Exp}_{\text{MS}, \mathcal{A}}^{\text{PK-CH}}(\kappa)$	
$pp \leftarrow_s \text{PGen}(1^\kappa); b \leftarrow_s \{0, 1\}; \rho \leftarrow_s \text{sample}_\rho(pp); (sk_1, pk_1) \leftarrow_s \text{KGen}(pp, \ell(\kappa))$	
$(sk_2^0, pk_2^0) \leftarrow_s \text{KGen}(pp, \ell(\kappa)); pk_2^1 \leftarrow \text{ConvertPK}(pk_1, \rho); sk_2^1 \leftarrow \text{ConvertSK}(sk_1, \rho)$	
$b' \leftarrow_s \mathcal{A}^{\text{Sign}(pp, sk_1, \cdot), \text{Sign}(pp, sk_2^b, \cdot)}(pk_1, pk_2^b); \text{return } b = b'$	

**Definition 5 (Origin-hiding).** A MS scheme is origin-hiding if for all  $\kappa, pp \in \text{PGen}(1^\kappa), pk^*, m$ , and  $\sigma$ , the following two properties hold:

1. if  $\text{Verify}(pk, M, \sigma) = 1$  and  $\mu \leftarrow_s \text{sample}_\mu$ , then  $\text{ChgRep}(pk^*, m, \sigma, \mu)$  outputs uniformly random  $M' \in [M]_{\mathcal{R}_M}$  and  $\sigma' \in \{\hat{\sigma} \mid \text{Verify}(pk^*, M', \hat{\sigma}) = 1\}$ .
2. if  $\text{Verify}(pk, M, \sigma) = 1$  and  $\rho \leftarrow_s \text{sample}_\rho$ , then  $\text{ConvertSig}(pk^*, M, \sigma, \rho)$  outputs a uniformly random  $\sigma' \in \{\hat{\sigma} \mid \text{Verify}(\text{ConvertPK}(pk^*, \rho), M, \hat{\sigma}) = 1\}$  and  $\text{ConvertPK}(pk^*, \rho)$  outputs a uniformly random element of  $[pk^*]_{\mathcal{R}_{pk}}$ .

In the following, we present the MS by Crites and Lysyanskaya [32], which is an extension of the EQS from [43]. It's the state-of-the-art signature in terms of efficiency and has its security proven in the generic group model for Type-III pairings. The message space is  $(\mathbb{G}^*)^\ell$  where  $\ell$  is the length of the message vector. We recall that all elements of a vector  $(M)_{i \in [\ell]} \in (\mathbb{G}^*)^\ell$  share different mutual ratios that depend on their discrete logarithms. Hence, it is possible to partition  $(\mathbb{G}^*)^\ell$  into equivalence classes given by:  $\mathcal{R} = \{(M, M') \in (\mathbb{G}^*)^\ell \times (\mathbb{G}^*)^\ell \mid \exists s \in \mathbb{Z}_p^* : M' = M^s\} \subseteq (\mathbb{G}^*)^\ell$ . Moreover, an analogous relation can be defined for the public keys, inducing equivalence classes on the key space as well.

$\text{PGen}(1^\kappa) \rightarrow \text{pp}$ : **return**  $\text{BGen}(1^\kappa)$ .  
 $\text{KGen}(\text{pp}, \ell) \rightarrow (\text{pk}, \text{sk})$ :  $\forall 1 \leq i \leq \ell : x_i \leftarrow_{\mathbb{S}} \mathbb{Z}_p^*$ ;  $\text{sk} \leftarrow (x_i)_{i \in [\ell]}$ ;  $\text{pk} \leftarrow (\hat{G}^{x_i})_{i \in [\ell]}$   
**return**  $(\text{pk}, \text{sk})$ .  
 $\text{Sign}(\text{pp}, \text{sk}, M) \rightarrow \sigma$ :  $y \leftarrow_{\mathbb{S}} \mathbb{Z}_p^*$ ;  $Z \leftarrow (\prod_{i=1}^{\ell} M_i^{x_i})^y$ ;  $Y \leftarrow G^{\frac{1}{y}}$ ;  $\hat{Y} \leftarrow \hat{G}^{\frac{1}{y}}$   
**return**  $(Z, Y, \hat{Y})$ .  
 $\text{Verify}(\text{pp}, M, \sigma, \text{pk} = (\hat{X}_i)_{i \in [\ell]}) \rightarrow 0/1$ :  
**return**  $\prod_{i=1}^{\ell} e(M_i, \hat{X}_i) = e(Z, \hat{Y}) \wedge e(Y, \hat{G}) = e(G, \hat{Y})$ .  
 $\text{ConvertSK}(\text{sk}, \rho) \rightarrow \text{sk}'$ :  $\text{sk}' \leftarrow \rho \cdot \text{sk}$ ; **return**  $\text{sk}'$ .  
 $\text{ConvertPK}(\text{pk}, \rho) \rightarrow \text{pk}'$ :  $\text{pk}' \leftarrow \text{pk}^\rho$ ; **return**  $\text{pk}'$ .  
 $\text{ConvertSig}(\text{pk}, M, \sigma, \rho) \rightarrow \sigma'$ :  $\psi \leftarrow_{\mathbb{S}} \mathbb{Z}_p^*$ ; **return**  $(Z^{\psi\rho}, Y^{\frac{1}{\psi}}, \hat{Y}^{\frac{1}{\psi}})$ .  
 $\text{ChgRep}(\text{pk}, M, \sigma, \mu) \rightarrow (M', \sigma')$ :  $\psi \leftarrow_{\mathbb{S}} \mathbb{Z}_p^*$ ;  $M' \leftarrow M^\mu$ ;  $\sigma' \leftarrow (Z^{\psi\mu}, Y^{\frac{1}{\psi}}, \hat{Y}^{\frac{1}{\psi}})$   
**return**  $(M', \sigma')$ .

**Theorem 1** ([32]). *The above MS scheme is unforgeable, public key class-hiding and origin-hiding in the generic group model for Type-III bilinear groups. Moreover, it is message class-hiding if the DDH assumption holds in  $\mathbb{G}$ .*

### 2.3 Verifiable Secret Sharing

Our construction from Sect. 5.2 uses the verifiable secret sharing scheme by Pedersen [55]. In this paper we follow the notation in [1]. Let  $G$  and  $H$  be two elements of  $\mathbb{G}$  s.t. the discrete logarithm of  $H$  with base  $G$  is unknown. To share a secret  $y$  in  $\mathbb{Z}_p$ , a dealer first chooses two  $t$ -degree random polynomials  $F_y(X)$  and  $D_y(X)$  from  $\mathbb{Z}_p[X]$  s.t.  $F_y(0) = y$ . Let  $R_y$  denote the random free term of  $D_y(X)$ . The dealer sends a pair  $(y^j, R_y^j) := (F_y(j), D_y(j))$  to party  $P_j$  via a private channel. Subsequently, it broadcasts  $EY^k := G^{a_k} H^{b_k}$  (a Pedersen commitment) for  $k = 0, \dots, t$  where  $a_k$  and  $b_k$  are the  $k$ -degree coefficients of  $F_y(X)$  and  $D_y(X)$  respectively. Given  $EY^k$ , correctness of a share  $(y^j, R_y^j)$  can be verified by checking  $G^{y^j} + H^{R_y^j} = \prod_{k=0}^t EY^{k^j}$ . Hereinafter, we denote the execution of this verifiable secret sharing protocol by  $\text{VSS}(y, R_y)[G, H] \xrightarrow{F_y, D_y} (y^j, R_y^j)[EY^0, EY^1, \dots, EY^t]$ .

## 3 Threshold Mercurial Signatures

We follow the notation from [29] to present the syntax and security properties.

**Definition 6 (Threshold mercurial signature)**. *A TMS scheme is a MS scheme where  $\text{KGen}$  and  $\text{Sign}$ , are replaced with:*

$\text{TKGen}(\text{pp}, \ell, t, n) \rightarrow (\vec{\text{sk}}, \vec{\text{pk}}, \text{pk})$ : *On input the public parameters  $\text{pp}$ , a length parameter  $\ell$ , and two integers  $t, n \in \text{poly}(1^\kappa)$  such that  $1 \leq t \leq n$ , this PPT algorithm outputs two vectors of size  $n$  of signing and public keys along with the global (threshold) public key  $\text{pk}$ . Both, the signing keys  $\vec{\text{sk}} = (\text{sk}_1, \dots, \text{sk}_n)$  and the public keys  $\vec{\text{pk}} = (\text{pk}_1, \dots, \text{pk}_n)$  are distributed among parties such that party  $P_i$  gets  $(\text{sk}_i, \vec{\text{pk}}, \text{pk})$ . The message space  $\mathcal{M}$  is well-defined from  $\text{pp}$  and  $\ell$ .*

$\text{TSign}(\text{pp}, \{\text{sk}_j\}_{j \in \mathcal{T}}, M) \rightarrow \sigma$ : On input  $\{\text{sk}_j\}_{j \in \mathcal{T}}$  for some  $\mathcal{T} \subseteq [n], |\mathcal{T}| \geq t$  and a message  $M \in \mathcal{M}$ , this PPT algorithm is run interactively by a set of parties in  $\mathcal{T}$ . At the end, they either abort or output a signature  $\sigma$ .

We also consider threshold key converter versions for shared keys ( $\text{ConvertTPK}$  and  $\text{ConvertTSK}$ ) that are analogous to  $\text{ConvertPK}$  and  $\text{ConvertSK}$  (now acting on the global keys). For convenience, we include an algorithm  $\text{SimTKGen}$  that given a key pair  $(\text{pk}, \text{sk})$ , on input  $\text{pk}$ ,  $n$  and a subset of corrupted parties  $\mathcal{C} \subsetneq [n]$  of size  $t - 1$ , outputs  $\mathcal{C}$  shares for  $\text{sk}$  and  $n$  shares of  $\text{pk}$  according to Definition 7.

*Security Properties.* A public key of our TMS consists of independent random group elements, and distributed  $\text{TKGen}$  can be instantiated with a DKG protocol. Since many DKG protocols are available with various security properties, e.g., [2, 25, 39, 54] (we also refer to a recent survey on the history and state of the art on DKG [47]), we consider construction of distributed  $\text{TKGen}$  an independent topic and consider  $\text{TKGen}$  being done by a single trusted party throughout the paper. Nevertheless, we state the security of  $\text{TKGen}$  required in this work as follows.

**Definition 7 (Security of key generation).**  $\text{TKGen}$  is secure if it outputs  $\text{pk}$  with the same distribution as  $\text{KGen}$  does, and there exists a simulator,  $\text{SimTKGen}$  that, for any sufficiently large  $\kappa$ , any  $\text{pp} \in \text{PGen}(1^\kappa)$ ,  $\ell \in \mathbb{N}$ ,  $(\text{pk}, \text{sk}) \in \text{KGen}(\text{pp}, \ell)$ ,  $t, n \in \mathbb{N}$ ,  $\mathcal{C} \subsetneq [n]$  of size  $t - 1$ ,  $\text{SimTKGen}(\text{pk}, n, \mathcal{C})$  outputs  $\{\text{sk}_j\}_{j \in \mathcal{C}}$  and  $\{\text{pk}_j\}_{j \in [n]}$ . The joint distribution of  $(\text{pk}, \{\text{pk}_j\}_{j \in [n]}, \{\text{sk}_j\}_{j \in \mathcal{C}})$  is indistinguishable from that of  $\text{TKGen}(\text{pp}, \ell, t, n)$ .

Correctness of TMS follows the usual notion for MS. Thus, we defer its formal definition to the full version of this work. For unforgeability and unlinkability (class-hiding) we follow the definitions from [29, 32], adapting them to the threshold setting (the adversary can corrupt up to  $t - 1$  parties). To that end, we consider a signing oracle  $\text{OTSign}$  that internally executes  $\text{TSign}$  as a step function in a sequentially interactive protocol where computations are executed by a party while obtaining input from the previous and passing the output to the next one. Consequently,  $\text{TSign}$  can be seen as a sequence of step functions  $\text{TSign}^j$  for  $j \in [k]$  where  $k$  is the total number of steps of the protocol, i.e.,  $k = 3(t - 2) + 2 \times 2 = 3t - 2$ . We define a function  $\phi : [k] \rightarrow [n]$  that maps the index of a step to the id of the party executing the step. On receiving a query  $(M, \mathcal{T})$ , oracle  $\text{OTSign}$  executes  $\text{TSign}^j$  for  $j = 1$  to  $k$  in sequence. If party  $\phi(j)$  is corrupted,  $\text{OTSign}$  consults adversary  $\mathcal{A}$  and obtains the output of  $\text{TSign}^j$ . Otherwise,  $\text{OTSign}$  executes  $\text{TSign}^j$  and sends the output to  $\mathcal{A}$ .  $\text{OTSign}$  finally outputs  $\sigma$  that  $\text{TSign}^k$  outputs. For ease of exposition, we define an auxiliary internal procedure ( $\text{Steps}()$ ) in  $\text{OTSign}$  to capture it.

**Definition 8 (Unforgeability).** A TMS scheme is unforgeable if the advantage of any PPT adversary  $\mathcal{A}$  defined by  $\text{Adv}_{\text{TMS}, \ell, t, n}^{\text{UNF}}(1^\kappa, \mathcal{A}) := \Pr[\text{Exp}_{\text{TMS}, \ell, t, n}^{\text{UNF}}(1^\kappa, \mathcal{A}) \Rightarrow \text{true}] \leq \epsilon(\kappa)$ , where  $\text{Exp}_{\text{TMS}, \ell, t, n}^{\text{UNF}}(1^\kappa, \mathcal{A})$  is shown in Fig. 1.

Unlike the original class-hiding definition for mercurial signatures (Definition 4), we aim to capture a stronger definition in which the adversary is given access

Experiment $\mathbf{Exp}_{\text{TMS},\ell,t,n}^{\text{UNF}}(1^\kappa, \mathcal{A})$	Steps(SK, $M$ )
$\text{st} \leftarrow \emptyset; \Sigma \leftarrow \emptyset; \text{pp} \leftarrow \text{\$ PGen}(1^\kappa); (\mathcal{C}, \text{st}) \leftarrow \mathcal{A}(\text{st}, \text{pp})$ <b>if</b> $\mathcal{C} \not\subseteq [n] \vee  \mathcal{C}  > t - 1$ <b>return</b> $\perp$ $\mathcal{H} \leftarrow [n] \setminus \mathcal{C}; (\vec{\text{sk}}, \vec{\text{pk}}, \text{pk}) \leftarrow \text{\$ TGen}(\text{pp}, \ell, t, n)$ $(M^*, \sigma^*, \rho^*) \leftarrow \text{\$ } \mathcal{A}^{\text{OTSign}(\text{sk}, \cdot, \cdot)}(\text{st}, \{\text{sk}_i\}_{i \in \mathcal{C}}, \vec{\text{pk}}, \text{pk})$ <b>return</b> $\forall M \in \Sigma : [M]_{\mathcal{R}_M} \neq [M^*]_{\mathcal{R}_M}$ $\wedge \text{Verify}(M^*, \sigma^*, \text{ConvertPK}(\text{pk}, \rho^*)) = 1$	$\text{steps} \leftarrow \emptyset$ <b>foreach</b> $j \in [3t - 2]$ <b>do</b> <b>if</b> $\phi(j) \in \mathcal{C}$ <b>then</b> $(\text{st}, \text{steps}) \leftarrow \mathcal{A}(\text{st}, \text{steps})$ <b>else</b> $\text{steps} \leftarrow \text{TSign}^j(\text{SK}, M; \text{steps})$ $\text{st} \leftarrow \mathcal{A}(\text{st}, \text{steps})$ <b>return</b> steps
Oracle $\text{OTSign}(\text{sk}, M, \mathcal{T})$ <b>if</b> $ \mathcal{T}  \neq t \vee \mathcal{T} \cap \mathcal{H} = \emptyset$ <b>return</b> $\perp$ $\sigma \leftarrow \text{Steps}(\{\vec{\text{sk}}_j\}_{j \in \mathcal{T}}, M); \Sigma \leftarrow \Sigma \cup \{M\};$ <b>return</b> $\sigma$	

**Fig. 1.** Unforgeability experiment.  $\mathcal{C}$  and  $\mathcal{H}$  are the sets of corrupt and honest signers.

to the challenge public keys and shares of the corresponding secret keys associated to corrupted parties for one of them. Our approach is to consider a scenario under *key leakage* where the adversary gets to know a subset of the secret key shares, similar to the class-hiding definition from [8]. Our notion is in-between the class-hiding notion from [32] that only considers honestly generated keys with no key leakage and the one from [8] (which is strictly weaker than  $(\cdot, 1, 3)$ -UNL from [22]). We refer to it as *public key unlinkability* to make a distinction. For  $n = t = 1$ , our definition seamlessly gives the notion of public key unlinkability for MS, implied by public key class-hiding and fulfilled by the instantiation of MS in [32]. We will use these facts to prove public key unlinkability.

**Definition 9 (Public Key Unlinkability).** *A TMS scheme is public key unlinkable if the advantage of any PPT adversary  $\mathcal{A}$  defined by  $\text{Adv}_{\text{TMS},\ell,t,n}^{\text{PK-UNL}}(1^\kappa, \mathcal{A}) := 2 \cdot \Pr[\mathbf{Exp}_{\text{TMS},\ell,t,n}^{\text{PK-UNL}}(1^\kappa, \mathcal{A}) \Rightarrow \text{true}] - 1 \leq \epsilon(\kappa)$ , where  $\mathbf{Exp}_{\text{TMS},\ell,t,n}^{\text{PK-UNL}}(1^\kappa, \mathcal{A})$  is shown in Fig. 2.*

*Remark 1.* We stress that the above definition serves two purposes. First, it provides backward compatibility with the original MS formalization, assuming the usual single-party signing process where the adversary cannot corrupt a party and thus is not given any secret key shares. Secondly, as discussed in the context of anonymous credentials (Sect. 4.4), when a signature is distributively computed, no adversary can distinguish if an adapted signature that verifies under an adapted public key is related or not to the original public key, even if it knows a subset of shares for the corresponding secret key. Before, since the adversary knew the full secret key, it was trivial to distinguish an adapted public key from one from a different equivalence class.



Experiment  $\mathbf{Exp}_{\text{TMS}, \ell, t, n}^{\text{PK-UNL}}(1^\kappa, \mathcal{A})$

---

$\text{st} \leftarrow \emptyset; b \leftarrow_{\$} \{0, 1\}; \rho \leftarrow_{\$} \text{sample}_\rho; \text{pp} \leftarrow_{\$} \text{PGen}(1^\kappa)$   
 $(\mathcal{C}, \text{st}) \leftarrow \mathcal{A}(\text{st}, \text{pp}); \text{if } \mathcal{C} \notin [n] \vee |\mathcal{C}| > t - 1 \text{ return } \perp$   
 $(\vec{\text{sk}}_1, \vec{\text{pk}}_1, \text{pk}_1) \leftarrow_{\$} \text{TKGen}(\text{pp}, \ell, t, n); (\vec{\text{sk}}_2^0, \vec{\text{pk}}_2^0, \text{pk}_2^0) \leftarrow_{\$} \text{TKGen}(\text{pp}, \ell, t, n)$   
 $\text{pk}_2^1 \leftarrow \text{ConvertPK}(\text{pk}_1, \rho); \vec{\text{sk}}_2^1 \leftarrow \text{ConvertSK}(\vec{\text{sk}}_1, \rho)$   
 $b' \leftarrow_{\$} \mathcal{A}^{\text{OTSign}(\{\vec{\text{sk}}^{(j)}\}_{j \in \mathcal{T}, \dots})}(\text{st}, \{\vec{\text{sk}}_1^{(j)}\}_{j \in \mathcal{C}}, \text{pk}_1, \text{pk}_2^b); \text{return } b = b'$   
 Oracle  $\text{OTSign}(M, \text{pk}, \mathcal{T})$

---

**if**  $|\mathcal{T}| \neq t$  **return**  $\perp$ ;  $\Sigma \leftarrow \Sigma \cup \{M\}$   
**if**  $\text{pk} = \text{pk}_2^b$  **return**  $\text{Sign}(\vec{\text{sk}}_2^b, M)$  **else if**  $\text{pk} = \text{pk}_1$  **return**  $\text{Steps}(\{\vec{\text{sk}}^{(j)}\}_{j \in \mathcal{T}}, M)$   
 $\text{Steps}(\text{SK}, M)$

---

$\text{steps} \leftarrow \emptyset$   
**foreach**  $j \in [3t - 2]$  **do**  
**if**  $\phi(j) \in \mathcal{C}$  **then**  $(\text{st}, \text{steps}) \leftarrow \mathcal{A}(\text{st}, \text{steps})$   
**else**  $\text{steps} \leftarrow \text{TSign}^j(\text{SK}, M; \text{steps}); \text{st} \leftarrow \mathcal{A}(\text{st}, \text{steps})$   
**return**  $\text{steps}$

**Fig. 2.** Public key unlinkability experiment.

## 4 Two-Party Case

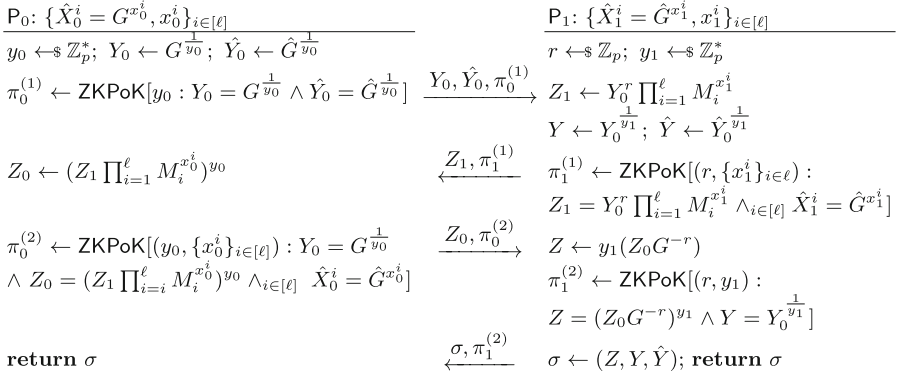
We present the two-party case as a (2-2)-TMS scheme. This decision will become clearer when we discuss the applications of this setting.

Our approach to building a (2-2)-TMS is to modify the scheme from [32] so that the signing protocol runs interactively between two parties. Intuitively, a signature that verifies under a jointly computed public key is obtained at the end. The resulting signature has the same structure as the one from [32] and it can work as a drop-in replacement. In particular, we assume one honest party for public key unlinkability (if they collude they can recognize the public key).

### 4.1 Construction 1

We assume that  $\text{PGen}(1^\kappa)$  and  $\text{TKGen}(\text{pp}, \ell, 2, 2)$  are run honestly. Every signer  $j$  is given  $\text{pp} = (p, \mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, G, \hat{G}, e)$ ,  $\vec{\text{pk}} := \{\hat{G}^{x_j^i}\}_{j \in [0,1]}^{i \in [\ell]}$ ,  $\text{pk}$ , and  $\vec{\text{sk}}_j := \{x_j^i\}_{i \in [\ell]}$ . The (global) signing key  $x_i$  for  $i \in [\ell]$  is implicitly set to  $x_i := x_0^i + x_1^i \in \mathbb{Z}_p$ .

In Fig. 3, we present our main protocol for instantiating  $\text{TSign}$ . The protocol's goal is to compute  $(Z, Y, \hat{Y})$  for a given message  $M$ . It consists of two parts; one to compute  $Y$  and  $\hat{Y}$ , and another to compute  $Z$ . Below we give an intuition and subsequently discuss the technical details required to prove security.



**Fig. 3.** TSign(pp,  $\{x_j^i\}_{j \in \{0,1\}, i \in [\ell]}, M$ )

Computing  $Y = G^{\frac{1}{y}}$  and  $\hat{Y} = \hat{G}^{\frac{1}{y}}$  for  $y = y_0 y_1$  is done straightforwardly in sequence. Computing  $Z = (\prod M_i^{x_0^i + x_1^i})^y$  for  $i \in [\ell]$  could be done first by computing  $Z_1 = \prod M_i^{x_1^i}$  at signer  $P_1$ , then  $Z_0 = Z_1 \prod M_i^{x_0^i}$  at signer  $P_0$ , and finally  $(Z_0)^{y_0 y_1}$  with  $y_0$  and  $y_1$  in sequence. However,  $Z_1$  is computed deterministically, requiring full knowledge about  $P_1$ 's signing key, while we have to simulate  $P_1$  without knowing the signing keys for the case where  $P_0$  is corrupted.

Our approach to getting around the above problem is to blind  $Z_1$  by using  $Y_0 = G^{\frac{1}{y_0}}$ , obtained in the first part of the protocol as the basis of a blinding factor. Computing  $Z_1 = Y_0^r \prod M_i^{x_1^i}$  with random  $r$  perfectly blinds it. Once  $P_0$  computes  $Z_0 = (Z_1 \prod M_i^{x_0^i})^{y_0}$ , factor  $Y_0$  in  $Z_1$  is cancelled out since  $(Y_0^r)^{y_0} = (G^{\frac{r}{y_0}})^{y_0} = G^r$ . Thus,  $P_1$ , who holds  $r$ , can easily unblind  $Z_0$  by multiplying  $G^{-r}$ . This blinding of  $Z_1$  causes another problem in the opposite case where  $P_1$  is corrupted; It makes it hard for the simulator to control the resulting signature. We address it by extracting the randomization factor  $r$  from the zero-knowledge proof of well-formedness of blinded  $Z_1$ . Since the unblinding is deterministic with respect to  $r$ , the simulator knowing  $r$  can embed an intended signature to  $Z_0$ .

As previously mentioned, we require zero-knowledge proofs to prove the right computation of the values sent by each party. In particular, we require knowledge soundness of  $\pi_1^{(1)}$  and zero-knowledge of  $\pi_0^{(1)}$  and  $\pi_0^{(2)}$  to simulate  $P_0$ . Analogously, to simulate  $P_1$ . Below we discuss how each ZKPoK can be implemented.

- $\pi_0^{(1)} := \text{ZKPoK}[y_0 : Y_0 = G^{1/y_0} \wedge \hat{Y}_0 = \hat{G}^{1/y_0}]$ : This can be done with the standard Chaum-Pedersen protocol [24] with witness  $1/y_0$  instead of  $y_0$ .
- $\pi_1^{(1)} := \text{ZKPoK}[(r, \{x_1^i\}_{i \in [\ell]}) : Z_1 = Y_0^r \prod_{i=1}^{\ell} M_i^{x_1^i} \wedge_{i \in [\ell]} \hat{X}_1^i = \hat{G}^{x_1^i}]$ : as above.
- $\pi_0^{(2)} := \text{ZKPoK}[(y_0, \{x_0^i\}_{i \in [\ell]}) : Z_0 = (Z_1 \prod_{i=1}^{\ell} M_i^{x_0^i})^{y_0} \wedge G = Y_0^{y_0} \wedge_{i \in [\ell]} \hat{X}_0^i = \hat{G}^{x_0^i}]$ : The first clause involves a witness product in the exponent. The proof must not expose intermediate value  $Z_1 \prod_{i=1}^{\ell} M_i^{x_0^i}$  to the verifier. We thus translate the statement to the following equivalent one:

**Table 1.** Costs of ZKPoK protocols. Computation counts number of exponentiations in the relevant groups without optimization for multi-base exponentiations.

Proof	Computation		Proof Size
	Prover	Verifier	
$\pi_0^{(1)}$	$1 \mathbb{G}  + 1 \hat{\mathbb{G}} $	$2 \mathbb{G}  + 2 \hat{\mathbb{G}} $	$1 \mathbb{G}  + 1 \hat{\mathbb{G}}  + 2 \mathbb{Z}_p $
$\pi_1^{(1)}$	$(\ell + 1) \mathbb{G}  + \ell \hat{\mathbb{G}} $	$(\ell + 2) \mathbb{G}  + 2\ell \hat{\mathbb{G}} $	$1 \mathbb{G}  + \ell \hat{\mathbb{G}}  + (\ell + 1) \mathbb{Z}_p $
$\pi_0^{(2)}$	$(\ell + 2) \mathbb{G}  + \ell \hat{\mathbb{G}} $	$(\ell + 4) \mathbb{G}  + 2\ell \hat{\mathbb{G}} $	$2 \mathbb{G}  + \ell \hat{\mathbb{G}}  + (\ell + 1) \mathbb{Z}_p $
$\pi_1^{(2)}$	$3 \mathbb{G} $	$5 \mathbb{G} $	$2 \mathbb{G}  + 3 \mathbb{Z}_p $

$$\pi_0^{(2)} := \text{ZKPoK} \left[ \left( \{x_0^i\}_{i \in [\ell]}, 1/y_0 \right) : \begin{array}{l} Z_1 = Z_0^{1/y_0} \prod_{i=1}^{\ell} M_i^{-x_0^i} \\ \wedge Y_0 = G^{1/y_0} \wedge_{i \in [\ell]} \hat{X}_0^i = \hat{G}^{x_0^i} \end{array} \right]$$

- $\pi_1^{(2)} := \text{ZKPoK}[(r, y_1) : Z = (Z_0 G^{-r})^{y_1} \wedge Y = Y_0^{1/y_1}]$ : This statement involves a witness product as well, and is translated into  $\pi_1^{(2)} := \text{ZKPoK}[(1/y_1, r) : Z_0 = Z^{1/y_1} G^r \wedge Y = Y_0^{1/y_1}]$ . It is worth noting that  $r$  is not guaranteed to be the same as the one used in  $\pi_1^{(1)}$  since the final output is accepted if it verifies as a signature. Nevertheless,  $\pi_1^{(2)}$  is needed for the proper link between the resulting signature and the inputs from honest  $P_0$ .

Each proof is verified by the respective recipient. The same for  $\sigma$ . If any verification fails, the party aborts and TSign outputs  $\perp$ .

## 4.2 Efficiency

Except for ZKPoK's, computation and communication complexity at each party are the same as those for the original MS. Party  $P_1$  has three extra exponentiations in  $\mathbb{G}$  for blinding and unblinding.

Table 1 presents the computational and communication costs of each ZKPoK when instantiating them using sigma protocols. Computation costs count the number of exponentiations in the respective groups, and the proof size is in terms of scalar values and group elements. We defer the full presentation of each protocol to the full version ([5], Appendix A). We note that  $\ell$  is usually instantiated for short vectors. Considering the applications,  $\ell = 2$  for blind signatures,  $\ell = 3$  for the basic attribute-based credential scheme from [43],  $\ell = 5$  considering revocation [35], and  $\ell = 7$  for adding auditability [27].

## 4.3 Security

Correctness is proven in the full version of this work [5]. Unforgeability is considered for static corruptions in the stand-alone execution model. We first explain our proof strategy and key technical points. In mercurial signatures [32], unforgeability (Definition 3) is proved by contradiction. If there were a PPT algorithm

that could break unforgeability through accessing the signing oracle, one could construct a reduction breaking the unforgeability of the base SPS-EQ [43].

For TMS, the security assurance of unforgeability slightly alters the one from [32]. Since we have an interactive signing protocol, we must prove that the adversary's advantage when interacting with TSign is no greater than its advantage in the original unforgeability game. Moreover, we give strong power to the adversary allowing it to run the (interactive) signing oracle on behalf of any corrupted party of its choice instead of just leaking the key share of its choice. Hence, care should be taken when instantiating the signing oracle from Definition 8 as it can be run between the adversary and the environment. We have three cases:

1.  $\mathcal{A}$  calls the oracle for two honest parties (honest signing). In this case, the environment runs the honest protocol, and it is easy to see that the  $\mathcal{A}$ 's advantage is the same as in the original game due to the zero-knowledge property of the ZKPoK's and the fact that the signatures computed by the interactive protocol are identically distributed as the signatures from [32].
2.  $\mathcal{A}$  controls  $P_0$ . We simulate  $P_1$  so that it ignores inputs from  $P_0$  and outputs signatures following the same distribution as in the original game.
3.  $\mathcal{A}$  controls  $P_1$ . We simulate  $P_0$  based on  $P_1$ 's knowledge extraction.

In either case, we show that the joint view of the adversary and the corrupted party is essentially the same as that of the adversary in the original unforgeability game of MS due to the security of the zero-knowledge proofs involved.

**Theorem 2 (Unforgeability).** *Construction 1 is unforgeable against static corruption of at most one party if TKGen is secure, all ZKPoK's are secure, and the original MS is unforgeable.*

*Proof.* Given access to adversary  $\mathcal{A}$  playing the unforgeability game against TMS as in Fig. 1, we construct a simulator that plays the role of the adversary in the unforgeability game against MS as in Definition 3. Let  $(\text{sk}, \text{pk}) := (\{x^i\}_{i \in [\ell]}, \{\hat{X}^i\}_{i \in [\ell]})$  be a key pair of MS generated by KGen(pp,  $\ell$ ). Given pp as input, the simulator first invokes  $\mathcal{A}$  and outputs  $\mathcal{C}$  obtained from  $\mathcal{A}$ . Here,  $\mathcal{C}$  is either 0 or 1 meaning  $P_0$  or  $P_1$  is corrupted, respectively. Then, given pk as input, the simulator executes SimTKGen(pk, 2,  $\mathcal{C}$ ) to obtain  $\text{sk}_j := \{x_j^i\}_{i \in [\ell]}$  for  $j \in \mathcal{C}$  and  $\text{pk}_j := \{\hat{X}_j^i\}_{i \in [\ell]}$  for  $j \in [n]$ . Shared signing keys  $\{x_j^i\}_{i \in [\ell]}$  for  $j \notin \mathcal{C}$  are not given to the simulator but implicitly set so that  $x_0^i + x_1^i = x^i$  holds. The simulator then invokes  $\mathcal{A}$  with  $\{\text{sk}_j\}_{j \in \mathcal{C}}$ ,  $\{\text{pk}_j\}_{j \in [n]}$ , and pk as input.

Recall that  $\mathcal{A}$  is allowed to make signing queries to OTSign that internally executes TSign in the presence of a corrupted party. Thus, the simulator has to simulate the honest party in TSign. Whenever  $\mathcal{A}$  queries message  $M$  to OTSign, the simulator forwards  $M$  to signing oracle Sign of MS and obtains signature  $(Z', Y', \hat{Y}')$ . From here, the simulator works along with the possible corruption scenarios. The first case considers corruption of  $P_0$ , as shown in Fig. 4. The case in which  $P_1$  is corrupted is shown in Fig. 5.

$$\begin{array}{l}
\underline{P_0: \text{sk}_0, \text{pk}_0, \text{pk}_1, M \text{ (corr.)}} \qquad \underline{P_1: \text{pk}_0, \text{pk}_1, M \text{ (sim. with Sign(sk, \cdot))}} \\
(Z', Y', \hat{Y}') \leftarrow \text{Sign}(\text{sk}, M) \\
(Y_0, \hat{Y}_0, \pi_0^{(1)}) \leftarrow \mathcal{A}(\text{st}) \quad \xrightarrow{Y_0, \hat{Y}_0, \pi_0^{(1)}} Z_1 \leftarrow_{\$} \mathbb{G}; Y \leftarrow Y'; \hat{Y} \leftarrow \hat{Y}' \\
(Z_0, \pi_0^{(2)}) \leftarrow \mathcal{A}(\text{st}, Z_1, \pi_1^{(1)}) \quad \xrightarrow{Z_1, \pi_1^{(1)}} \pi_1^{(1)} \leftarrow \text{ZKPoK.Sim}(Z_1, Y_0, M) \\
\qquad \qquad \qquad \xrightarrow{Z_0, \pi_0^{(2)}} Z \leftarrow Z' \\
\qquad \qquad \qquad \qquad \qquad \qquad \qquad \qquad \pi_1^{(2)} \leftarrow \text{ZKPoK.Sim}(Z, Z_0, Y, Y_0) \\
\mathbf{return} (\sigma, \pi_1^{(2)}) \quad \xrightarrow{\sigma, \pi_1^{(2)}} \sigma \leftarrow (Z, Y, \hat{Y}); \mathbf{return} (\sigma, \pi_1^{(2)})
\end{array}$$

**Fig. 4.** Simulator's algorithm considering corruption of  $P_0$ .

$$\begin{array}{l}
\underline{P_0: \text{pk}_0, \text{pk}_1, M \text{ (sim. with Sign(sk, \cdot))}} \qquad \underline{P_1: \text{sk}_1, \text{pk}_0, \text{pk}_1, M \text{ (corr.)}} \\
(Z', Y', \hat{Y}') \leftarrow \text{Sign}(\text{sk}, M) \\
Y_0 \leftarrow Y'; \hat{Y}_0 \leftarrow \hat{Y}' \\
\pi_0^{(1)} \leftarrow \text{ZKPoK.Sim}(Y_0, \hat{Y}_0) \quad \xrightarrow{Y_0, \hat{Y}_0, \pi_0^{(1)}} (Z_1, \pi_1^{(1)}) \leftarrow \mathcal{A}(\text{st}, Y_0, \hat{Y}_0, \pi_0^{(1)}) \\
\qquad \qquad \qquad \xrightarrow{Z_1, \pi_1^{(1)}} \\
r \leftarrow \text{ZKPoK.Ext}(\pi_1^{(1)}); Z_0 \leftarrow Z' G^r \\
\pi_0^{(2)} \leftarrow \text{ZKPoK.Sim}(Z_0, Z_1, M, Y_0) \quad \xrightarrow{Z_0, \pi_0^{(2)}} (\sigma, \pi_1^{(2)}) \leftarrow \mathcal{A}(\text{st}, Z_0, \pi_0^{(2)}) \\
\mathbf{return} (\sigma, \pi_1^{(2)}) \quad \xrightarrow{\sigma, \pi_1^{(2)}} \mathbf{return} (\sigma, \pi_1^{(2)})
\end{array}$$

**Fig. 5.** Simulator's algorithm considering corruption of  $P_1$ .

We show that, for both cases of corruption, the honest party can be simulated indistinguishably from the real execution of the corresponding algorithm in  $\text{TSig}$ . For the first case (Fig. 4), the real computation of  $Z_1$  and the simulated one in the first round are perfectly indistinguishable because the real one includes a uniformly random factor and the simulated one is chosen uniformly. It implicitly determines random factor  $r := \log_{Y_0} Z_1 (\prod_{i=1}^{\ell} M_i^{x_i^1})^{-1}$  for  $x_i^1$  also implicitly determined by  $\hat{X}_1^1$ . Furthermore, the quality of simulated  $\pi_1^{(1)}$  is due to its zero-knowledge property. Moving into the second round, we claim that  $P_0$  cannot distinguish the difference between the original computation of  $\sigma$  and the simulated one provided that both  $\pi_0^{(1)}$  and  $\pi_0^{(2)}$  are sound. Observe that the proper computation of  $(Z, Y, \hat{Y})$  is deterministic from  $Z_0$ ,  $r$  and  $y_1$  implicitly determined by  $y_1 = \log_Y Y_0 = \log_{\hat{Y}} \hat{Y}_0$ . Therefore, if  $\pi_0^{(1)}$  and  $\pi_0^{(2)}$  are sound and  $\text{Sign}$  is correct, the simulated  $(Z, Y, \hat{Y})$  distributes perfectly in the same way as the original one does. The quality of simulated  $\pi_1^{(2)}$  is due to its zero-knowledge property, hiding how  $Z$  was computed.

We now consider corruption of  $P_1$  (Fig. 5). During the first round,  $Y_0$  and  $\hat{Y}_0$  are distributed identically as their original computation and  $\pi_0^{(1)}$  is zero-

knowledge. Looking at the second round, we claim that  $Z_0$  is perfectly simulated if  $\pi_1^{(1)}$  is knowledge sound. That is, the knowledge soundness of  $\pi_1^{(1)}$  assures that  $Z_1$  has been correctly computed as  $Z_1 = Y_0^r \prod_{i=1}^{\ell} M_i^{x_1^i}$  with the extracted  $r$ . Thus, for  $Z' = (\prod_{i=1}^{\ell} M_i^{x_i})^y$  where  $x_i = (x_0^i + x_1^i)$  for  $i \in [\ell]$  and  $y = y_0 = \log_G Y'$ , we have:  $Z_0 = Z'G^r = (\prod_{i=1}^{\ell} M_i^{x_0^i + x_1^i})^{y_0} G^r = (Y_0^r \prod_{i=1}^{\ell} M_i^{x_1^i})^{y_0} (\prod_{i=1}^{\ell} M_i^{x_0^i})^{y_0} = (Z_1 \prod_{i=1}^{\ell} M_i^{x_0^i})^{y_0}$ . Accordingly, the simulation of  $P_0$  is perfect modulo the knowledge soundness of  $\pi_1^{(1)}$  and zero-knowledge of  $\pi_0^{(1)}$  and  $\pi_0^{(2)}$ .

The simulator outputs whatever  $\mathcal{A}$  outputs. As  $\text{TKGen}$  is assumed secure and the honest party within  $\text{OTSign}$  is correctly simulated, the view of  $\mathcal{A}^{\text{OTSign}}$  is indistinguishable from that of the real unforgeability game in the presence of corrupt party  $\mathcal{C}$ . Thus, whenever  $\mathcal{A}$  is successful in forging TMS, so is the simulator in forging MS. Finally, we evaluate the advantage based on the error bounds for the sub-procedures. Let maximum error bounds for the zero-knowledge part be:  $\mathcal{E}_{\text{Snd}}$  and  $\mathcal{E}_{\text{ZK}}$  for the soundness and zero-knowledge of each proof, respectively, and  $\mathcal{E}_{\text{KSnd}}$  for the knowledge soundness of  $\pi_1^{(1)}$ . Also let  $\mathcal{E}_{\text{TKG}}$  denote the error bound for  $\text{SimTKGen}$ . We obtain that the adversary's advantage when executing  $q$  queries to the signing oracle is given by  $\text{Adv}_{\text{TMS}, \ell, 2, 2}^{\text{UNF}}(1^\kappa, \mathcal{A}) < \text{Adv}_{\text{MS}, \ell}^{\text{UNF}}(1^\kappa, \mathcal{A}) + q \cdot (2\mathcal{E}_{\text{Snd}} + 2\mathcal{E}_{\text{ZK}} + \mathcal{E}_{\text{KSnd}}) + \mathcal{E}_{\text{TKG}}$ .  $\square$

We prove unlinkability assuming at least one honest signer and consider a signing oracle in the presence of the corrupted party.

**Theorem 3 (Public Key Unlinkability).** *Construction 1 is public key unlinkable against static corruption of at most one party if  $\text{TKGen}$  is secure, all ZKPoK's are secure, and MS is origin and public key class-hiding.*

*Proof.* The proof strategy considers a similar simulator from Theorem 2. Given access to adversary  $\mathcal{A}$  playing the unlinkability game against TMS, we construct a simulator that plays the role of the adversary in the unlinkability game against MS (public key class-hiding). Given  $\text{pp}$  as input, the simulator invokes  $\mathcal{A}$  and outputs  $\mathcal{C}$  obtained from  $\mathcal{A}$ . Then, given  $(\text{pk}_1, \text{pk}_2^b)$  as input, the simulator runs  $\text{SimTKGen}$  for  $\text{pk}_1$  with  $\mathcal{C}$  to obtain, for  $\text{sk}_1^{(j)}$  for  $j \in \mathcal{C}$  and  $\text{pk}_1^{(j)}$  for  $j \in [n]$ . The simulator then invokes  $\mathcal{A}$  with  $\{\text{sk}_1^{(j)}\}_{j \in \mathcal{C}}$ ,  $\{\text{pk}_1^{(j)}\}_{j \in [n]}$ , and  $\text{pk}_2^b$  as input. The validity of the simulation up to this point is due to the security of  $\text{TKGen}$ .

On receiving a query from  $\mathcal{A}$  to  $\text{OTSign}$  on  $\text{pk}_2^b$  and  $M$ , the simulator forwards it to its oracle and returns the response to  $\mathcal{A}$ . This part of the simulation is perfect due to the origin-hiding property of MS.

On receiving a query from  $\mathcal{A}$  to  $\text{OTSign}$  on  $\text{pk}_1$  and  $M$ , the simulator forwards it to its oracle to obtain an MS signature for  $M$ . Subsequently, it uses the obtained signature to simulate an invocation of  $\text{TSign}$  (using Steps) on  $\text{pk}_1$  with  $M$  as explained in the proof of Theorem 2. Validity of this part of the simulation is due to the security of ZKPoK's as before. If Steps's simulation results in  $\perp$  (due to misbehaviour of a corrupted party), it returns  $\perp$ . Finally, the simulator outputs  $b'$  that  $\mathcal{A}$  outputs. Since the view of  $\mathcal{A}$  is correctly simulated, the output is correct whenever  $\mathcal{A}$  wins the game against TMS. This concludes the proof.  $\square$

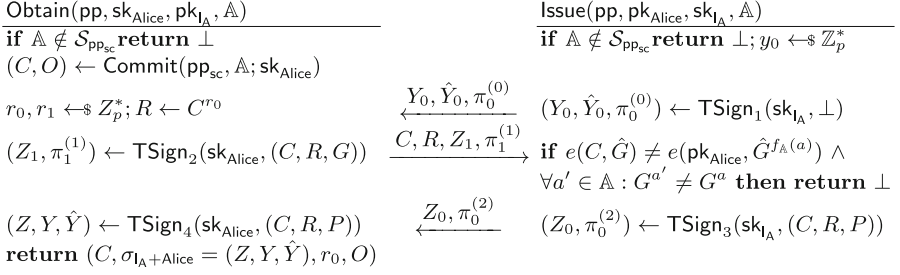
#### 4.4 Application to Anonymous Credentials

Attribute-based anonymous credentials (ABC) allow users to authenticate themselves with respect to a set of attributes while hiding their identity. A prominent framework in this setting based on EQS originated with the work by Fuchsbauer, Hanser and Slamanig [43] (hereinafter FHS19). FHS19 provides constant-size credentials (*i.e.*, the credential size as well as the bandwidth required to show it are constant-size irrespectively of the attributes shown) supporting a selective-disclosure of attributes (users can decide which attributes to show each time).

Subsequent works extended FHS19 to consider revocation [35], auditability [27], more expressiveness, and issuer-hiding features [15, 28]. In particular, [27] proposes to use the MS from [32] to provide issuer-hiding features based on an OR-Proof for the correct randomization of the issuer’s public key relative to a list of authorized issuers. Recall that using MS, users can adapt their signature to a randomized public key. A downside of this approach is that now the showing is linear in the number of issuers because of the OR-Proof. Furthermore, since the MS used only provides a weak issuer-hiding feature (*i.e.*, an issuer can recognize randomizations of its own public key), the ABC from [27] is limited to settings where partial trust can be tolerated. Put differently, a verifier colluding with an issuer can determine if the authenticated user belongs to the issuer’s organization, severely reducing the user’s anonymity set.

**A Closer Look at FHS19’s Framework.** The ABC from FHS19 enables anonymous and unlinkable credential showings. In terms of security, *unforgeability* of their credential scheme ensures that users can only authenticate with respect to attributes they possess. Besides, it also prevents any collusion among malicious users to collectively perform valid showings for attributes that none of them hold. *Anonymity* assures users that neither a verifier nor a malicious organization can collude to identify them. Additionally, it ensures that multiple showings of the same credential cannot be linked together (*i.e.*, credentials are multi-show). To realize an efficient ABC, FHS19 uses EQS on set-commitments where the latter primitive is basically an accumulator supporting subset membership proofs. Moreover, its public parameters are given by  $(G^{a^i}, G^{a^i})_{i \in [t]}$ , where  $t$  is the maximum cardinality of an attribute set (to be accumulated) and  $a$  is randomly picked and used as the accumulator’s evaluation point.

Each user produces a set commitment  $C$  to her set of attributes  $\mathbb{A} \subset \mathbb{Z}_p$ , whose randomness is the user’s secret key.  $\mathcal{S}_{\text{pp}_{\text{sc}}} = \{\mathbb{S} \subset \mathbb{Z}_p : 0 < |\mathbb{S}| \leq t\}$  defines the set of valid attributes sets and  $f_{\mathbb{S}}(X) := \prod_{s \in \mathbb{S}} (X - s)$  its polynomial representation, allowing everyone to efficiently compute  $G^{f_{\mathbb{A}}(a)}$  using  $G^{a^i}$  without knowledge of  $a$  itself. Following the original notation, a set commitment scheme includes a **Commit** algorithm that takes as input public parameters  $\text{pp}_{\text{sc}}$ , a set of attributes  $\mathbb{A}$ , and the user’s secret key as randomness. It returns a commitment  $C$  to the user’s attributes and opening information  $O$ . To obtain a credential, users interact with the issuer in the following way: they compute a commitment  $C$  to their attribute set and request a signature on  $(C, C^{r_0}, G)$ . The second and third elements in  $(C, C^{r_0}, G)$  are required to prove the scheme’s unforgeability.



**Fig. 6.** FHS19’s credential issuing protocol instantiated with our TMS.

**Improving Issuer-Hiding.** Our idea is to replace the signing protocol used in FHS19 to issue credentials with our interactive signing protocol for TMS. To that end, looking at Fig. 3, the user will play the role of  $P_1$  while the issuer will play the role of  $P_0$ . Moreover, since only the user needs to obtain the signature, we can remove the last step from  $P_1$ . In Fig. 6, we show how to instantiate the original issuing protocol from [43] (Fig. 2) considering a user Alice who runs  $\text{TSign}$  with an issuer  $\text{I}_A$ . For simplicity, we split  $\text{TSign}$  into  $\text{TSign}_i$  for  $i \in \{1, 2, 3, 4\}$  where each  $i$  abstracts the computation corresponding to the  $i$ -th round. As a result, we obtain a three-round protocol when using non-interactive ZKPoK instantiations.

Unlike the original protocol, in our case Alice obtains a signature  $\sigma_{\text{I}_A + \text{Alice}}$ , which verifies under  $\text{pk} = \text{pk}_{\text{I}_A} \cdot \text{pk}_{\text{Alice}}$ . Subsequently, Alice can produce a showing proof randomizing the signature-message pair with  $\mu$  (using  $\text{ChgRep}$  as in FHS19),  $\rho$  (using  $\text{ConvertSig}$  to hide  $\text{pk}$ ), and giving a NIZK proof for statement  $\{(\rho, \text{sk}_{\text{Alice}}) : \text{pk}' = (\text{pk}_{\text{I}_A} \cdot \text{pk}_{\text{Alice}})^\rho\}$ . Such type of NIZK, whose idea we borrow from [59], can be efficiently implemented in the ROM from Schnorr proofs. Intuitively, it attest that Alice generated her credential with the authority and thus the signature is valid. Note that Alice could produce the NIZK proof without having the TMS but that alone is useless. While this approach attest correctness, it is not yet issuer-hiding because it leaks the issuer’s public key  $\text{pk}_{\text{I}_A}$ . The user can generate an OR-Proof for the same previous statement for every key in the issuers’ set to make it (fully) issuer-hiding. Now, thanks to the public key unlinkability of TMS, issuers cannot link their public key with a randomized one (this was possible in all previous works from EQS [27]).

**Comparison with [59].** The recent work by Sanders and Traoré [59] provides a strong issuer-hiding notion with different trade-offs compared to our approach. A showing proof for [59] will be shorter when the attribute set is small and the use of PS signatures provides richer functionalities in terms of expressiveness (*e.g.*, one can easily prove relations between attributes). However, their showing is linear in the number of attributes encoded in a credential (ours is linear in the number of issuers), and the policies that are defined by each verifier to authorize a set of issuers are also linear in the number of attributes and the number of issuers. Besides, we provide backwards compatibility with all of the prior work



under the same framework, obtaining an ABC scheme with all its extensions (some of which are not considered in [59]).

## 5 Threshold Case

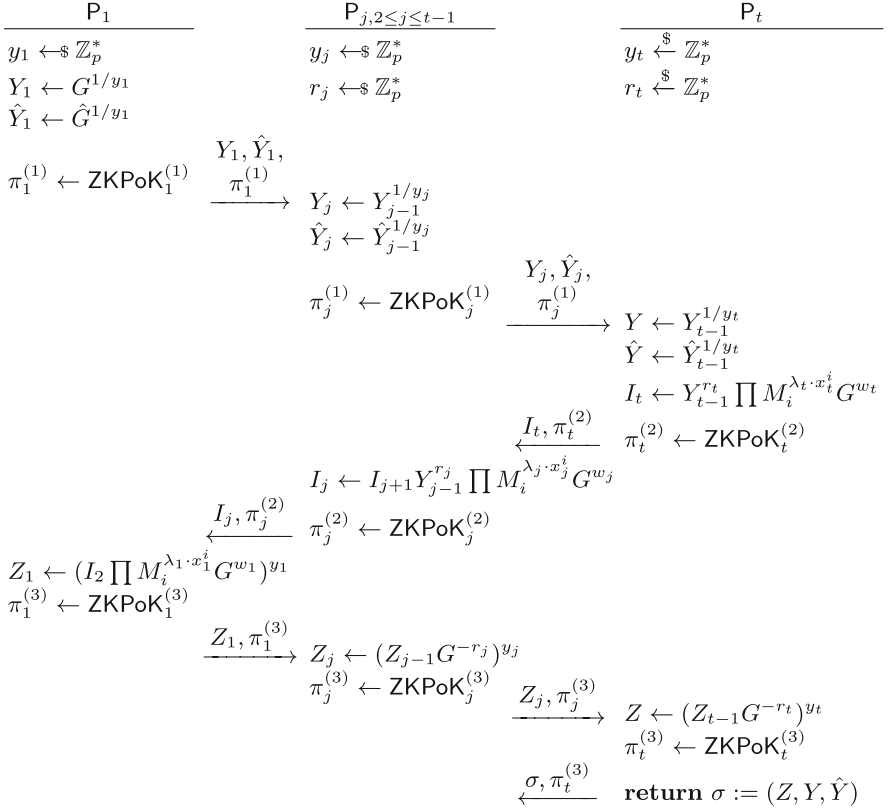
This section describes  $n$ -party protocols where keys are distributed in a  $(t, n)$ -threshold manner among  $n$  users  $P_1, \dots, P_n$ . Consequently, we focus on threshold signing protocols where a subset of users  $P_1, \dots, P_t$  engage to produce a signature. We assume the key generation is done by a single honest party and only describe the syntax and relation satisfied by the keys. Concretely, for given  $t$  and  $n$  such that  $1 \leq t \leq n$ ,  $\text{TKGen}$  generates local key pairs  $\text{sk}_j := (x_j^1, \dots, x_j^\ell)$ ,  $\text{pk}_j := (\hat{G}^{x_j^1}, \dots, \hat{G}^{x_j^\ell})$  for  $j \in [n]$ , and global public key. The global signing key is implicitly set to  $\text{sk} := (x_1, \dots, x_\ell)$  where each  $x_i$  is shared into  $(x_1^i, \dots, x_n^i)$  by  $(t, n)$ -threshold scheme over  $\mathbb{Z}_p$ . For any set of indices,  $\mathcal{T} \subseteq [n]$ , of size  $t$ , it holds that  $x_i = \sum_{j \in \mathcal{T}} \lambda_j x_j^i \pmod p$  where  $\lambda_j$  is a Lagrange coefficient defined as  $\lambda_j := \prod_{t \in \mathcal{T} \setminus \{j\}} \frac{t}{t-j} \pmod p$ .

### 5.1 Construction 2

Our first protocol follows the structure of the two-party case; parties work in sequence, from  $P_1$  to  $P_t$  communicating only over the public broadcast channel. The first  $P_1$  and the last  $P_t$  do the same as  $P_0$  and  $P_1$  did in the two-party case, respectively. However, each intermediate participant,  $P_2, \dots, P_{t-1}$ , must independently take on the roles of  $P_0$  and  $P_1$ , bringing forth new considerations to both the protocol and its security analysis. We present our construction in the preprocessing model where participating parties join the preprocessing procedure and then engage in the main signing protocol.

*Zero-Sharing over Public Channel.* In the preprocessing phase, participating parties set up shares of zero. We do this via Pedersen's commitments, and thus, extend the  $\text{pp}$  by adding  $H \in \mathbb{G}$  ( $H \neq G$ ) so that  $\text{PPGen}$  outputs  $\text{pp} = (\mathbb{G}, \hat{\mathbb{G}}, \mathbb{G}_T, H, G, \hat{G}, e)$ . At the beginning, each party  $P_j$  picks  $r_j \leftarrow_{\mathcal{S}} \mathbb{Z}_p^*$ , which is shared into  $r_{ji}$  so that  $P_j$  broadcasts  $r_{ji}, t_{ji}$  for all  $i \neq j$  and a commitment  $T_{ii} := G^{r_i} H^{t_i}$  of  $i = j$ . Commitments  $T_{ij} := G^{r_{ij}} H^{t_{ij}}$  for  $i \neq j$  are computed publicly. Thereafter, each  $P_j$  locally computes its share of zero as  $w_j := r_j - \sum r_{ij}$  and commits to it computing  $W_j := G^{w_j} H^{s_j}$ . It also computes  $t'_j := \sum t_{ji}$ . The last steps consists in broadcasting a NIKZ proof for the statement  $\pi_j := [w_j, s_j, t'_j : W_j = G^{w_j} H^{s_j} \wedge \prod T_{ji} = G^{w_j} H^{t'_j}]$ , whose verification confirms correctness of all shares.

*Protocol Description.* Without loss of generality, let  $\mathcal{T} = (1, \dots, t)$ , i.e.,  $(P_1, \dots, P_t) = (1, \dots, t)$  be the parties engaging in  $\text{TSign}(\text{pp}, \{\text{sk}_j\}_{j \in \mathcal{T}}, M)$  as presented in Fig. 7. Fully general description is recoverable by replacing  $\lambda_j \cdot x_j^i$  with  $\lambda_{P_j} \cdot x_{P_j}^i$  in the following. We follow the template of the two-party case, which operates



**Fig. 7.**  $t$ -party protocol for  $\text{TSign}(\text{pp}, \{\text{sk}_j\}_{j \in J}, M)$ . Products are taken for  $i = 1$  to  $\ell$ .

sequentially. The initial party  $P_1$  communicates with the first intermediate party  $P_2$ , and all intermediate parties behave the same until the last one,  $P_{t-1}$ , communicates with the final party  $P_t$ . The protocol proceeds backward until  $P_1$  is reached. Subsequently,  $P_1$  triggers the last round, which concludes when  $P_t$  broadcasts the signature. All proofs and the resulting signature are received and verified by everyone. If any party rejects, the output of the protocol is defined as  $\perp$ . Zero-knowledge proofs in Fig. 7 are defined as follows:

- $\pi_j^{(1)} := \text{ZKPoK}_j^{(1)}[y_j : Y_j = Y_{j-1}^{1/y_j} \wedge \hat{Y}_j = \hat{Y}_{j-1}^{1/y_j}]$  where  $Y_0 = G$  and  $\hat{Y}_0 = \hat{G}$  at  $j = 1$ .
- $\pi_j^{(2)} := \text{ZKPoK}_j^{(2)}[(r_j, w_j, s_j, \{x_j^i\}_{i \in [\ell]} : I_j = I_{j+1} \cdot Y_{j-1}^{r_j} \prod_{i=1}^{\ell} M_i^{\lambda_j \cdot x_j^i} G^{w_j} \wedge W_j = G^{w_j} H^{s_j} \wedge_{i \in [\ell]} \hat{X}_j^i = \hat{G}^{x_j^i}]$  where  $I_{t+1} = 1$  at  $j = t$ .
- $\pi_1^{(3)} := \text{ZKPoK}_1^{(3)}[(\{x_1^i\}_{i \in [\ell]}, y_1, w_1, s_1) : Z_1 = (I_2 \cdot \prod_{i=1}^{\ell} M_i^{\lambda_1 \cdot x_1^i} G^{w_1})^{y_1} \wedge Y_1^{y_1} = G \wedge W_1 = G^{w_1} H^{s_1} \wedge_{i \in [\ell]} X_1^i = \hat{G}^{x_1^i}]$
- $\pi_j^{(3)} := \text{ZKPoK}_j^{(3)}[y_j : Z_j = (Z_{j-1} \cdot G^{-r_j})^{y_j} \wedge Y_j^{y_j} = Y_{j-1}]$

An obvious difference from the two-party case is the presence of the intermediate parties,  $P_2, \dots, P_{t-1}$ . Computing  $Y = G^{\frac{1}{y}}$  and  $\hat{Y} = \hat{G}^{\frac{1}{y}}$  is done sequentially from  $P_1$  to  $P_t$ , and  $y$  is defined by  $y = \prod_{j=1}^t y_j$ . If all parties are honest, the following holds for  $Z_1$  (recall that at  $Z_1$  all  $G^{w_j}$  are cancelled out):

$$Z_1 = (I_2 \cdot \prod_{i=1}^{\ell} M_i^{\lambda_1 \cdot x_1^i})^{y_1} = (G^{\frac{r_2}{y_1} + \frac{r_3}{y_2 y_1} + \dots + \frac{r_t}{y_{t-1} \dots y_1}} \cdot \prod_{i=1}^{\ell} M_i^{\sum_{j=1}^t \lambda_j \cdot x_j^i})^{y_1} \quad (1)$$

The reason why  $Z_1$  is computed in the second stage is the same as the two-party case: the blinding is useful for constructing the simulator in the presence of corrupted parties. It allows computing  $Z$  by sequentially unblinding  $Z_1$  in the reverse order from  $P_2$  to  $P_t$ . To see that the blinding factors are canceled as expected, observe that

$$\begin{aligned} Z &= (Z_{t-1} \cdot G^{-r_t})^{y_t} = (\dots (Z_1 \cdot G^{-r_2})^{y_2} \dots)^{y_{t-1}} \cdot G^{-r_t})^{y_t} \\ &= (\dots (G^{\frac{r_2}{y_1} + \frac{r_3}{y_2 y_1} + \dots + \frac{r_t}{y_{t-1} \dots y_1}} \cdot \prod_{i=1}^{\ell} M_i^{\sum_{j=1}^t \lambda_j \cdot x_j^i})^{y_1} G^{-r_2})^{y_2} \dots)^{y_{t-1}} \cdot G^{-r_t})^{y_t} \end{aligned}$$

holds. Concerning the exponent of  $G$ , we have:

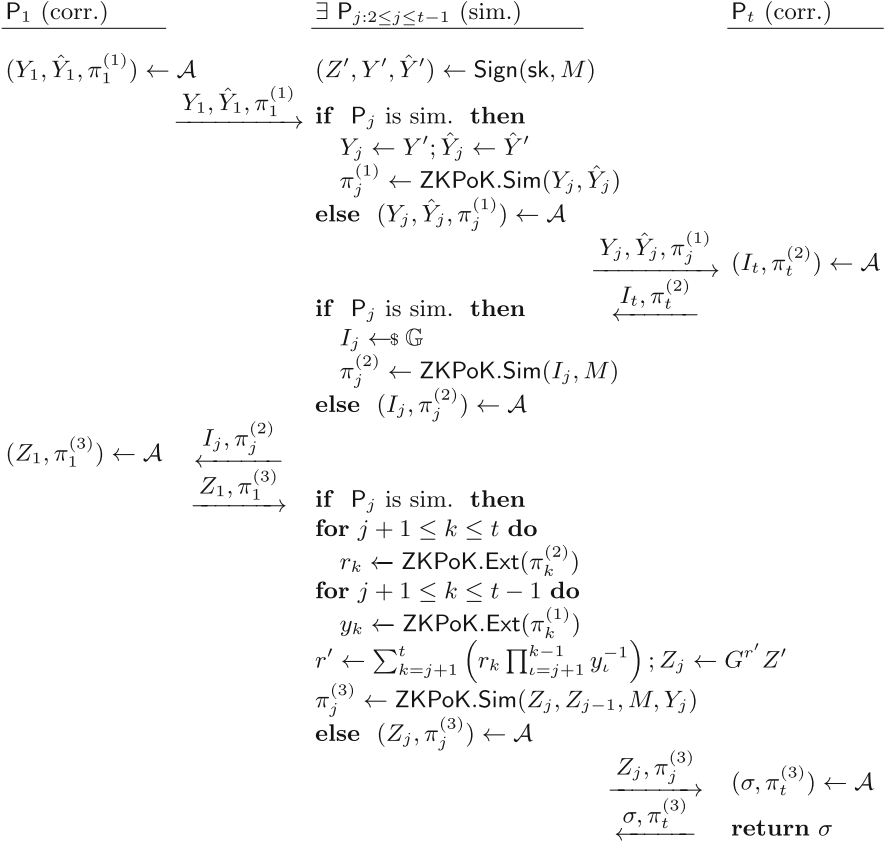
$$\begin{aligned} &(\dots (\frac{r_2}{y_1} + \frac{r_3}{y_2 y_1} + \dots + \frac{r_t}{y_{t-1} \dots y_1}) y_1 - r_2) \dots) y_{t-1} - r_t) y_t \\ &= (\dots (\frac{r_t}{y_{t-1}} + r_{t-1}) \frac{1}{y_{t-2}} \dots) \frac{1}{y_3} + r_3) \frac{1}{y_2} + r_2) \frac{1}{y_1} \cdot y_1 - r_2) \dots) y_{t-1} - r_t) y_t \\ &= (0 + \dots + 0) y_t = 0. \end{aligned}$$

Therefore:  $Z = (\prod_{i=1}^{\ell} M_i^{\sum_{j=1}^t \lambda_j \cdot x_j^i})^{\prod_{j=1}^t y_j} = (\prod_{i=1}^{\ell} M_i^{x_i})^y$ .

**Efficiency.** We first note that the two-move pre-processing phase can be interleaved with the interactive signing protocol from Fig. 7 for a total of 5 moves. Besides, computation at this stage is cheap and most of the communication involves transmitting elements in  $\mathbb{Z}_p$ . The ZKPoK's are analogous to the two-party case as shown in the full version ([5], Appendix A). Communication and computation increase linearly with  $t$ . In many cases, however, the number of signers does not grow beyond one order of magnitude so  $t$  can stay relatively small. Furthermore, considering other applications such as multi-signatures and threshold ring signatures,  $\ell$  will be small, meaning the ZKPoK's will be efficient and short.

**Security.** The key observation for correctness was given above. The strategies to prove unforgeability and unlinkability are similar to their two-party counterparts. An essential difference, however, is the presence of intermediate parties to simulate, considering a situation in which at most  $t-1$  signers can be corrupted.

**Theorem 4 (Unforgeability).** *Construction 2 is unforgeable against static corruption of at most  $t-1$  parties if TKGen is secure, all ZKPoK's are adaptive zero-knowledge, simulation extractable, and the original MS is unforgeable.*



**Fig. 8.** Simulator for intermediate party  $P_j$ .  $\mathcal{A}$  manages its internal state. At  $k = j+1$ , product  $\prod_{\ell=j+1}^{k-1} y_\ell^{-1}$  is defined as 1.

*Proof.* As the overall proof structure is the same as the two-party case, we focus on describing the simulation of the signing oracle in the presence of corrupted parties. We begin by considering the simplest case where there is only one honest party among the participating parties,  $P_1, \dots, P_t$ . Depending on the position of the player, the simulation differs as follows:

- If the initial party,  $P_1$ , is the honest one, it is simulated similarly as simulating  $P_0$  in the two-party case considering the random factor in an accumulated form. Precisely, in the first step, it sets  $(Y_1, \hat{Y}_1) := (Y', \hat{Y}')$  and simulates  $\pi_1^{(1)}$ . In the second step, it extracts  $r_j$  and  $y_j$  from all other parties and computes their accumulated random factor

$$r' := r_2 + \frac{r_3}{y_2} + \dots + \frac{r_t}{y_{t-1} \cdots y_2} = \sum_{k=2}^t \left( r_k \prod_{\ell=2}^{k-1} y_\ell^{-1} \right)$$

where product  $\prod_{l=2}^{k-1} y_l^{-1}$  is defined as 1 at  $k = 2$ . (Note that  $y_t$  is unnecessary.) It then outputs  $Z_1 = G^{r'} Z'$  and simulates  $\pi_1^{(3)}$ . Provided that  $\pi_j^{(1)}$  and  $\pi_j^{(2)}$  for  $j > 1$  allow knowledge extraction,  $Z_1$  distributes the same as in the real protocol run. Indeed, as we argued for correctness (see Eq. (1)), if all parties are honest,  $Z_1 = (G^{\frac{r_2}{y_1} + \frac{r_3}{y_2 y_1} + \dots + \frac{r_t}{y_{t-1} \dots y_1}} \cdot \prod_{i=1}^{\ell} M_i^{\sum_{j=1}^t \lambda_j \cdot x_j^i})_{y_1} = G^{r_2 + \frac{r_3}{y_2} + \dots + \frac{r_t}{y_{t-1} \dots y_2}} (\prod_{i=1}^{\ell} M_i^{\sum_{j=1}^t \lambda_j \cdot x_j^i})_{y_1} = G^{r'} Z'$  where  $y_1$  is implicitly determined by  $1/\log_G Y_1$ . Thus, the output from  $P_1$  is perfectly simulated modulo the simulation extractability errors of  $\pi_2^{(1)}, \dots, \pi_{t-1}^{(1)}$  and  $\pi_t^{(2)}, \dots, \pi_2^{(2)}$ , and zero-knowledge of  $\pi_1^{(1)}$  and  $\pi_1^{(3)}$ .

- If the tail party,  $P_t$ , is the honest one, the simulation is the same as that of simulating  $P_1$  in the two-party case except for the obvious notational adjustment. In the first step, it sets  $(Y, \hat{Y})$  by  $(Y', \hat{Y}')$ , samples  $I_t$  uniformly from  $\mathbb{G}$ , and simulates  $\pi_t^{(2)}$ . In the second step, it sets  $Z := Z'$  and simulates  $\pi_t^{(3)}$ . The simulation is perfect modulo the soundness errors of  $\pi_1^{(1)}, \dots, \pi_{t-1}^{(1)}$ ,  $\pi_{t-1}^{(2)}, \dots, \pi_2^{(2)}$ ,  $\pi_1^{(3)}, \dots, \pi_{t-1}^{(3)}$ , (that assure that  $(Y_{t-1}, \hat{Y}_{t-1})$  and  $(Y, \hat{Y})$  are in the same distribution), and zero-knowledge of  $\pi_t^{(2)}$  and  $\pi_t^{(3)}$ .
- If an intermediate party,  $P_j$ ,  $j \in [2, t-1]$ , is the honest one, the simulation is done as shown in Fig. 8. It is a mixture of the above simulation strategies. It works like the initial party for the right (ascending) parties in the first and third steps, and like the tail party for the left (descending) parties in the second step. For the same reason for above cases, the simulation in this case is perfect modulo the soundness errors of  $\pi_1^{(1)}, \dots, \pi_{j-1}^{(1)}$ ,  $\pi_j^{(2)}, \dots, \pi_1^{(2)}$ , and  $\pi_1^{(3)}, \dots, \pi_{j-1}^{(3)}$ , and simulation extractability errors of  $\pi_{j+1}^{(1)}, \dots, \pi_{t-1}^{(1)}$  and  $\pi_t^{(2)}, \dots, \pi_{j+1}^{(2)}$ , and zero-knowledge of  $\pi_j^{(1)}$ ,  $\pi_j^{(2)}$ , and  $\pi_j^{(3)}$ .

The above procedure relies on the proof  $\pi^{(1)}$  and  $\pi^{(2)}$  for simulation and extraction, which requires simulation extractability. For the simulation extractability error ( $\mathcal{E}_{\text{SimExt}}$ ), the soundness errors ( $\mathcal{E}_{\text{Snd}}$ ), and zero-knowledge errors ( $\mathcal{E}_{\text{ZK}}$ ) we obtain that the adversary's advantage when executing  $q$  queries to the signing oracle is given by  $\mathbf{Adv}_{\text{TMS}, \ell, t, n}^{\text{UNF}}(1^\kappa, \mathcal{A}) < \mathbf{Adv}_{\text{MS}, \ell}^{\text{UNF}}(1^\kappa, \mathcal{A}) + q((t-1)(2\mathcal{E}_{\text{SimExt}} + 2\mathcal{E}_{\text{Snd}}) + 3\mathcal{E}_{\text{ZK}})$ . This concludes the case where only one honest party is participating.

As the next step, we consider the case where only two parties,  $P_u$  and  $P_v$  for  $u < v$ , are honest and all others are corrupted. The simulation is as follows:

- Simulate the right-hand honest party,  $P_v$ , in the same way as the single honest party case as above.
- For the left-hand honest party,  $P_u$ , first follow the protocol to output  $Y_u, \hat{Y}_u$ , and  $\pi_u^{(1)}$  in the first round. Then, in the second round, pick  $I_u$  uniformly from  $\mathbb{G}$  and simulate  $\pi_u^{(2)}$ . Finally, in the third round, pick  $Z_u$  uniformly from  $\mathbb{G}$  and simulate  $\pi_u^{(3)}$ .

We claim that the simulation is perfect modulo the simulations of the relevant zero-knowledge proofs, which we ignore in the following. We follow the game transition argument as follows:

- Let  $\theta_0$  be a joint view of corrupted parties in a protocol run.
- We first replace the right-hand honest party,  $P_v$ , with the above simulation. Let  $\theta_1$  be the joint view obtained during the protocol. Since the single honest party case is the perfect simulation,  $\theta_0$  and  $\theta_1$  distributes identically.
- Next we replace the left-hand honest party,  $P_u$ , with the simulation. Let the resulting view as  $\theta_2$ . We claim that  $\theta_1$  and  $\theta_2$  distribute identically and prove it in the full version of this work ([5], Claim 1).

From the above, we conclude the two-honest-party case. Since the simulation is perfect, it straightforwardly extends to the most general case where an arbitrary number of honest parties are present in the signing process:

- Simulate the rightmost honest party as done for the single honest party case.
- For each of other left-located honest parties, follow the simulation procedure of the left-hand honest party in the two honest party case.

To show that the simulation remains perfect, we follow the same procedure where we first replace the rightmost party with the simulation, and then replace other honest parties one by one in the descending order, *i.e.*, from the right to left. In every transition, the output from the simulated party distributes identically to the original one. This concludes the proof of Theorem 4.  $\square$

The following theorem can be proven similarly to the two-party case.

**Theorem 5 (Public Key Unlinkability)** *Construction 2 is public key unlinkable against static corruption of at most  $t - 1$  parties if TKGen is secure, all ZKPoK's are secure, and the original MS is origin and public key class-hiding.*

*Proof.* We proceed as in Theorem 3 except that we make use of the simulation strategies addressed in the above proof of unforgeability (Theorem 4). We construct a simulator that plays the role of the adversary in the unlinkability game against MS given access to an adversary  $\mathcal{A}$  who plays the unlinkability game against TMS. As before, given  $\text{pp}$  as input, the simulator invokes  $\mathcal{A}$  and outputs  $\mathcal{C}$  obtained from  $\mathcal{A}$ . Then, given  $(\text{pk}_1, \text{pk}_2^b)$  as input, the simulator runs SimTKGen for  $\text{pk}_1$  with  $\mathcal{C}$  to obtain, for  $\text{sk}_1^{(j)}$  for  $j \in \mathcal{C}$  and  $\text{pk}_1^{(j)}$  for  $j \in \mathcal{T}$ . The simulator then invokes  $\mathcal{A}$  with  $\{\text{sk}_1^{(j)}\}_{j \in \mathcal{C}}$ ,  $\{\text{pk}_1^{(j)}\}_{j \in \mathcal{T}}$ , and  $\text{pk}_2^b$  as input. Again, validity of the simulation up to this point is due to the security of TKGen.

On receiving a query from  $\mathcal{A}$  to OTSign on  $\text{pk}_2^b$  and  $M$ , the simulator forwards it to its oracle and returns the response to  $\mathcal{A}$ . This part of the simulation is perfect due to the origin-hiding property of MS.

On receiving a query from  $\mathcal{A}$  to OTSign on  $\text{pk}_1$  and  $M$ , the simulator forwards it to its oracle to obtain an MS signature for  $M$ . Subsequently, it uses the obtained signature to simulate an invocation of TSign (using Steps) on  $\text{pk}_1$  with  $M$  as explained in the proof of Theorem 4. Validity of this part of the simulation is due to the security of ZKPoK's as before. If Steps's simulation results in  $\perp$  (due to misbehavior of a corrupted party), it returns  $\perp$ .

The simulator outputs whatever  $\mathcal{A}$  outputs. Since the view of  $\mathcal{A}$  is correctly simulated, the output is correct whenever  $\mathcal{A}$  wins the game against TMS.  $\square$

### 5.2 Construction 3

Our second protocol builds on more general results and thus follows a different strategy compared to the previous constructions. Specifically, we observe that the (non-interactive) distributed multiplication protocol proposed by Abe [1] aligns well with the signing requirements. This protocol enables parties to compute the product of  $x$  and  $y$  (the key operation in the MS signing algorithm) using the Pedersen VSS scheme presented in Sect. 2.3. Abe’s protocol tolerates up to  $n/2$  corrupt players, resulting in a four-move interactive signing process that eliminates the need for any zero-knowledge proofs as we describe next.

As before, we let  $\mathcal{T} = (1, \dots, t)$ , i.e.,  $(P_1, \dots, P_t) = (1, \dots, t)$ . Recall that our goal is to compute  $Z = \prod_{i=1}^{\ell} M_i^{x_i y}$ ,  $Y = G^{\frac{1}{y}}$  and  $\hat{Y} = \hat{G}^{\frac{1}{y}}$ . This can be done by having each party deliver verifiable shares  $w_i^{jk} := x_i^{jk} y^{jk}$ , allowing to reconstruct the share of product  $x_i^k y^k$  without revealing any hidden shares. It remains to compute the final  $\frac{1}{y}$  but this becomes cumbersome given only shares of  $y$ . Our approach is the use of an auxiliary value  $C$  (whose shares  $C^j$  are also distributed via VSS) that can be used to compute  $Cy$  easily following Abe’s method. In brief, any party that gets  $Cy \in \mathbb{Z}_p$  can compute  $d^j := (Cy)^{-1} C^j$  from the values they hold. This translates into a share for  $\frac{1}{y}$  that can restore  $\frac{1}{y}$  by Lagrange interpolation. Finally, upon obtaining  $\{x_i y\}_{i \in [\ell]}$  and  $\frac{1}{y}$ , parties can compute their partial signatures and gather them to output the full signature as shown below (analogous for  $\hat{Y}$ ):

$$\begin{aligned} Z &= \prod_{j \in \mathcal{T}} Z^j = \prod_{i=1}^{\ell} M_i^{\sum_{j \in \mathcal{T}} \lambda_j w_i^j} = \prod_{i=1}^{\ell} M_i^{x_i y} = \left( \prod_{i=1}^{\ell} M_i^{x_i} \right)^y \\ Y &= \prod_{j \in \mathcal{T}} Y^j = G^{\sum_{j \in \mathcal{T}} \lambda_j d^j} = G^{\sum_{j \in \mathcal{T}} (Cy)^{-1} \lambda_j C^j} = G^{\frac{C}{Cy}} = G^{\frac{1}{y}} \end{aligned} \tag{2}$$

Successively, recall that TKGen generates local keys  $\text{sk}_j := (x_j^1, \dots, x_j^{\ell})$  and  $\text{pk}_j := (\hat{G}^{x_j^1}, \dots, \hat{G}^{x_j^{\ell}})$  for  $j \in \mathcal{T}$ , and global public key  $\text{pk}$ . As before, the global signing key is implicitly set to  $\text{sk} := (x_1, \dots, x_{\ell})$  where each  $x_i$  is shared into  $(x_1^i, \dots, x_n^i)$  using  $(t, n)$ -threshold scheme over  $\mathbb{Z}_p$ . This can be done with Pedersen’s VSS to obtain  $(\{x_j^i\}_{j \in \mathcal{T}}^{i \in [\ell]}, \{\hat{G}^{x_j^i}\}_{j \in \mathcal{T}}^{i \in [\ell]}, \text{pk}) \leftarrow \text{TKGen}(\text{pp}, \ell, t, n)$  such that for each  $x_i$  a unique random value  $R_{x_i} \xleftarrow{\$} \mathbb{Z}_p$  is computed from polynomials  $F_{x_i}(X)$  and  $D_{x_i}(X)$  alongside the corresponding commitments as shown below:

$$\text{VSS}(x_i, R_{x_i})[G, H] \xrightarrow{F_{x_i}, D_{x_i}} (\{x_j^i\}, \{R_{x_i}^j\})_{j \in \mathcal{T}}^{i \in [\ell]} [EX_i^0, EX_i^1, \dots, EX_i^n] \tag{3}$$

As a result, every party  $P_j$  obtains  $(\{x_j^i\}_{i \in [\ell]}, \{\hat{G}^{x_j^i}\}_{i \in [\ell]}, \{R_{x_i}^j, EX_i^j\}_{i \in [\ell]}^{j \in \mathcal{T}}, \text{pk})$ , where each  $EX_i^j$  is needed to verify the validity of the shares  $x_j^i$  and  $R_{x_i}^j$ . The next step is to do the same for the value  $Y$ :

$$\text{VSS}(y, R_y)[G, H] \xrightarrow{F_y, D_y} (y^j, R_y^j) [EY^0, EY^1, \dots, EY^n]$$

**Table 2.** Costs of construction 2. Computation counts number of exponentiations in the relevant groups without optimization for multi-base exponentiations.

MPC Round	Computation	Communication
Round I - $P_j$	-	$6t \mathbb{G}  + 5(\ell + 1)$
Round II - $P_k$	$6(\ell + 1)(t + 3) \mathbb{G} $	$\ell + 1$
Round III - $P_j$	$(\ell + 1) \mathbb{G}  +  \hat{\mathbb{G}} $	$2 \mathbb{G}  +  \hat{\mathbb{G}} $
Round IV - $P_k$	$2 \mathbb{G}  +  \hat{\mathbb{G}} $	-

For ease of exposition, we also assume a dealer who computes the VSS for  $y$  and  $C$ . Once that all parties have shares for  $x$  and  $y$ , Abe’s protocol proceeds with each  $P_j$  picking a  $t$ -degree random polynomial to share  $x_j y_j$ . The share  $C^{j_i}$  is privately sent to party  $i$ .

The resulting non-interactive protocol  $\text{TSign}(\{\text{sk}_j\}_{j \in \mathcal{T}}, M)$  for  $M = (M_1, \dots, M_\ell)$  as run by parties  $P_j$  ( $1 \leq j \leq t$ ) is given in Fig. 9. We denote by  $VY^j$  the product  $\prod_{m=0}^t EY^{mj^m}$  and  $\langle \dots \rangle$  is used for variables that every party can compute locally. We observe that parties  $P_j, k$  can be selected in any way. When  $P_j$  prepares the shares for  $P_k$ ,  $w_i^{jk}$  denotes  $x_i^{jk} y^{jk}$  and  $v^{jk}$  denotes  $C^{jk} y^{jk}$ . In addition,  $\mathcal{K}$  is the subgroup of  $[\ell]$  including enough number to restore  $Cy$  and  $\mathcal{T}$  is the subgroup of  $n$  including more than  $t - 1$  numbers to compute the full signature. Security of this protocol directly follows from that of the underlying VSS and Abe’s multiplication protocol. We analyse its efficiency on Table 5.2.

### 5.3 Application to Delegatable Credentials

The first work on delegatable anonymous credentials from MS by Crites and Lysyanskaya [32] was based on the following idea: a root authority (or CA) produces a signature on Alice’s public key, who can subsequently do the same to delegate her credential to Bob. More in detail, the CA produces a signature  $\sigma_{\text{CA} \rightarrow \text{Alice}}$  for a message  $M = \text{pk}_{\text{Alice}}$ , which Alice can use to authenticate herself by proving knowledge of  $\text{sk}_{\text{Alice}}$ . To delegate her credential, Alice signs Bob’s public key  $\text{pk}_{\text{Bob}}$ , producing  $\sigma_{\text{Alice} \rightarrow \text{Bob}}$ . This approach works for any regular signature scheme. However, as observed in [32], when the signatures are MS, they can be adapted to provide stronger privacy guarantees. In brief, if Alice is known to Bob under a pseudonymous public key  $\text{pk}_{\text{Alice}'}$ , she can consistently adapt the signatures in the delegation chain to produce  $\sigma_{\text{CA} \rightarrow \text{Alice}'}$  and  $\sigma_{\text{Alice}' \rightarrow \text{Bob}}$ . This way, Bob can prove knowledge of his secret key for a valid chain (*i.e.*, a chain that goes all the way back to a valid signature from the CA).

Subsequent work by Crites and Lysyanskaya [33] and Putman and Martin [58] improved the original construction to support attributes. In all cases, it is essential that the CA’s signature is a MS so that it can be adapted to a new user pseudonym (even if the root key is never randomized). Because of this, all previous work assume a single trusted issuer as the CA. Our TMS construction can be used as a drop-in replacement for the CA’s signature to distribute trust.



**Dealer :**

$R_y, R_c \xleftarrow{\$} \mathbb{Z}_p$  and generate  $F_y, D_y, F_c$  and  $D_c$ .  
 $\text{VSS}(y, R_y)[G, H] \xrightarrow{F_y, D_y} (y^j, R_y^j)[EY^0, EY^1, \dots, EY^n]$   
 $\text{VSS}(C, R_C)[G, H] \xrightarrow{F_C, D_C} (C^j, R_C^j)[EC^0, EC^1, \dots, EC^m]$   
**return** to  $P_j$  ( $1 \leq j \leq t$ ) :  $(y_j, R_y^j, \{EY^j\}_{j \in \mathcal{T}}, C^j, R_C^j, \{EC^j\}_{j \in \mathcal{T}})$

**Party  $j$  :**  $(M, \text{sk}_j, R_j, y^j, R_y^j, C^j, R_C^j)$

**for**  $i \in [\ell]$ :

$R_{xy}^j \xleftarrow{\$} \mathbb{Z}_p$  and generate  $F_{xy}, D_{xy}, F_{xy}^\dagger, D_{xy}^\dagger, F_{xy}^{\dagger\dagger}$  and  $D_{xy}^{\dagger\dagger}$ .  
 $\text{VSS}(x_i^j, R_{x_i}^j)[G, H] \xrightarrow{F_{xy}, D_{xy}} (x_i^{jk}, R_{x_i}^{jk})[\langle VX_i^{jk} \rangle, EX_i^{j1}, \dots, EX_i^{jt}]$   
 $\text{VSS}(x_i^j, R_{x_i}^j)[VY^j, H] \xrightarrow{F_{xy}^\dagger, D_{xy}^\dagger} (\langle x_i^{jk} \rangle, R_{x_i}^{jk})[EXY_i^{j0}, EXY_i^{j1}, \dots, EXY_i^{jt}]$   
 $\text{VSS}(x_i^j \cdot y^j, R_y^j \cdot x_i^j + R_{xy}^j)[G, H] \xrightarrow{F_{xy}^{\dagger\dagger}, D_{xy}^{\dagger\dagger}} (w_i^{jk}, R_{w_i}^{jk})[\langle EXY_i^{jk} \rangle, EW_i^{j1}, \dots, EW_i^{jt}]$   
 $R_{Cy}^j \xleftarrow{\$} \mathbb{Z}_p$ ; Generate  $F_{Cy}, D_{Cy}, F_{Cy}^\dagger, D_{Cy}^\dagger, F_{Cy}^{\dagger\dagger}$  and  $D_{Cy}^{\dagger\dagger}$ .  
 $\text{VSS}(C^j, R_C^j)[G, H] \xrightarrow{F_{Cy}, D_{Cy}} (C^{jk}, R_C^{jk})[\langle VC^j \rangle, EC^{j1}, \dots, EC^{jt}]$   
 $\text{VSS}(C^j, R_C^j)[VC^j, H] \xrightarrow{F_{Cy}^\dagger, D_{Cy}^\dagger} (\langle C^{jk} \rangle, R_C^{jk})[ECY^{j0}, ECY^{j1}, \dots, ECY^{jt}]$   
 $\text{VSS}(C^j \cdot y^j, R_y^j \cdot C^j + R_{Cy}^j)[G, H] \xrightarrow{F_{xy}^{\dagger\dagger}, D_{xy}^{\dagger\dagger}} (v^{jk}, R_v^{jk})[\langle ECY^{j0} \rangle, EV^{j1}, \dots, EV^{jt}]$   
**return** to  $P_k$  ( $1 \leq k \leq t$ ):

$(\{x_i^{jk}\}_{i \in \ell}, \{R_{x_i}^{jk}\}_{i \in \ell}, \{R_{xy}^{jk}\}_{i \in \ell}, \{w_i^{jk}\}_{i \in \ell}, \{R_{w_i}^{jk}\}_{i \in \ell}, C^{jk}, R_C^{jk}, R_{Cy}^{jk}, v^{jk}, R_v^{jk})$

**broadcast:**

$\{EX_i^{jk}\}_{k \in [t]}, \{EXY_i^{jk}\}_{k \in [t]}, \{EW_i^{jk}\}_{k \in [t]}, \{EC_i^{jk}\}_{k \in [t]}, \{ECY_i^{jk}\}_{k \in [t]}, \{EV_i^{jk}\}_{k \in [t]}$

**Party  $k$  :**  $(\dots)$

\*Verifies  $\{\{x_i^{jk}\}, \{R_{x_i}^{jk}\}, \{R_{xy}^{jk}\}, \{w_i^{jk}\}, \{R_{w_i}^{jk}\}\}_{i \in \ell}$

**for**  $i \in \ell$

**if**  $G^{x_i^{jk}} H^{R_{x_i}^{jk}} = VX_i^j \prod_{m=1}^t (EX_i^{jm})^{k^m} \wedge VY_j^{x_i^{jk}} H^{R_{xy}^{jk}} = \prod_{m=0}^t (EXY_i^{jm})^{k^m} \wedge$   
 $G^{w_i^{jk}} H^{R_{w_i}^{jk}} = EXY_i^{j0} \prod_{m=1}^t (EW_i^{jm})^{k^m}$  **then**  $w_i^k := \sum_{j \in \mathcal{T}} \lambda_j w_i^{jk}$  **else return**  $\perp$   
**if**  $G^{C^{jk}} H^{R_C^{jk}} = VC^j \prod_{m=1}^t (EC^{jm})^{k^m} \wedge VY_j^{C^{jk}} H^{R_{Cy}^{jk}} = \prod_{m=0}^t (ECY_i^{jm})^{k^m} \wedge$   
 $G^{v^{jk}} H^{R_v^{jk}} = ECY^{j0} \prod_{m=1}^t (EV^{jm})^{k^m}$  **then**  $v^k := \sum_{j \in \mathcal{T}} \lambda_j v^{jk}$  **else return**  $\perp$   
**return** to  $P_j$ :  $(\{w_i^k\}_{i \in [\ell]}, v^k)$

**Party  $j$  :**  $(\{\{w_i^k\}_{i \in [\ell]}\}_{k \in \mathcal{K}}, \{v^k\}_{k \in \mathcal{K}})$

$C_y := \sum_{k \in \mathcal{K}} \lambda_k v^k; d^j := (C_y)^{-1} C^j (= [y^{-1}]_j)$

Let  $Z^j := \prod_{i=1}^l M_i^{\lambda_j w_i^j}; Y^j := G^{\lambda_j d^j}; \hat{Y}^j := \hat{G}^{\lambda_j d^j}$  **return**  $(Z^j, Y^j, \hat{Y}^j)$  to  $P_k$

**Party  $k$  :**  $(\{Z^j\}_{j \in \mathcal{T}}, \{Y^j\}_{j \in \mathcal{T}}, \{\hat{Y}^j\}_{j \in \mathcal{T}})$

$Z := \prod_{j \in \mathcal{T}} Z^j; Y := \prod_{j \in \mathcal{T}} Y^j; \hat{Y} := \prod_{j \in \mathcal{T}} \hat{Y}^j$ ; **return**  $\sigma := (Z, Y, \hat{Y})$

**Fig. 9.** Non-interactive TSign

With this in mind, any threshold of authorized issuers could produce a signature on the user’s public key, enabling more use cases for this primitive as in the case where users get credentials from certain government authorities discussed in [32].

## 6 Experimental Evaluation

We prototyped our (interactive) constructions in Rust based on the mercurial signature implementation from [27]. However, we replaced the BLS12-381 crate by Filecoin’s BLS12-381 crate (blasters [49], a Rust wrapper around the blst library [48]). Our implementation and related documentation is available in [52]. As previously mentioned, we only considered cases where  $\ell \in \{2, 5, 10\}$ , which cover all known applications. We also implemented our schemes switching the message and public key groups. However, since the ZKPoK’s require parties to prove knowledge of their secret key when computing the multi-exponentiations to the message part, no significant change in performance is gained. Nonetheless, if one relaxes the security requirement for semi-honest parties, switching groups would improve performance at the cost of a slightly bigger signature size (elements in  $\mathbb{G}_1$  and  $\mathbb{G}_2$  are of size 48 and 96 bytes, respectively).

Table 3 summarizes the execution times for the signing algorithm of our TMS and the original MS. Verification times are the same for all variants (1.8ms for  $\ell = 2$ , 3ms for  $\ell = 5$  and 5ms for  $\ell = 10$ ). We used the nightly compiler, the Criterion library, and all the benchmarks were run on a MacBook Pro M3 with 32 GB of RAM with no extra optimizations. In all cases, the standard deviation was below 1ms. For TMS we considered the two-party and threshold cases with five and ten parties. As expected, the computational complexity of our interactive signing process scales linearly with the number of parties. This is also the case for communication. More concretely, the initial and final parties broadcast two ZKPoK’s and receive  $3n - 4$ . Similarly, intermediate parties broadcast three messages and receive  $3n - 5$ . Nevertheless, considering that all the applications discussed require a few parties, and the additional features offered by TMS, we find the overhead compared to standard MS well justified.

The most closely related work to ours is the threshold SPS from [29], but, to the best of our knowledge, it has not been implemented. This leave us with few options to compare the performance of TMS. One option could be to consider the multi-signature MuSig2 from [53], but their work is incomparable to ours as it works in a pairing-free group and focuses on other functionalities.

**Table 3.** Timing of signature generation for messages of size  $\ell$  in milliseconds.

Scheme	# of Parties	Signing (ms)		
		$\ell = 2$	$\ell = 5$	$\ell = 10$
MS [43]	1	0.3	0.4	0.5
TMS (Sect. 4)	2	3.9	6.2	10.1
TMS (Sect. 5.1)	5	13.3	19.3	29.6
TMS (Sect. 5.1)	10	28.0	40.8	60.5

## 7 Conclusion

In this work, we develop interactive threshold mercurial signatures (TMS). To showcase the power of this primitive, we presented constructions for both the two-party and multi-party cases, discussing their instantiation under different scenarios. Our experimental evaluation suggests that our constructions are practical when instantiated in the ROM. Most importantly, our interactive approach allows us to generate signatures with an affine linear transformation in the public key structure, translating into stronger privacy properties for many applications. Something that previous works in the setting were unable to achieve.

We also explored the instantiation of TMS from standard building blocks, finding it practical. Compared to the existing threshold structure-preserving signatures for the generalized case, our constructions are very competitive in terms of efficiency while covering other use cases.

All in all, interactive TMS offer greater flexibility than standard MS with relatively little overhead. Therefore, revising existing applications of MS (and more in general EQS) through the optics of TMS can be a very promising direction for future work. Another is the study of alternatives that could offer straight-line knowledge extraction to obtain concurrently secure schemes.

**Acknowledgements.** The authors thank the anonymous reviewers of Asiacrypt 2024 for their insightful comments and very helpful suggestions.

## References

1. Abe, M.: Robust distributed multiplication without interaction. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 130–147. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999). [https://doi.org/10.1007/3-540-48405-1\\_9](https://doi.org/10.1007/3-540-48405-1_9)
2. ABe, M., Fehr, S.: Adaptively secure feldman VSS and applications to universally-composable threshold cryptography. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 317–334. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 2004). [https://doi.org/10.1007/978-3-540-28628-8\\_20](https://doi.org/10.1007/978-3-540-28628-8_20)
3. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 2010). [https://doi.org/10.1007/978-3-642-14623-7\\_12](https://doi.org/10.1007/978-3-642-14623-7_12)
4. Abe, M., Groth, J., Haralambiev, K., Ohkubo, M.: Optimal structure-preserving signatures in asymmetric bilinear groups. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 649–666. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2011). [https://doi.org/10.1007/978-3-642-22792-9\\_37](https://doi.org/10.1007/978-3-642-22792-9_37)
5. Abe, M., Nanri, M., Perez Kempner, O., Tibouchi, M.: Interactive threshold mercurial signatures and applications. Cryptology ePrint Archive, Paper 2024/625 (2024), <https://eprint.iacr.org/2024/625>

6. Alper, H.K., Burdges, J.: Two-round trip schnorr multi-signatures via delinearized witnesses. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 157–188. Springer, Heidelberg, Germany, Virtual Event (Aug 16–20, 2021). [https://doi.org/10.1007/978-3-030-84242-0\\_7](https://doi.org/10.1007/978-3-030-84242-0_7)
7. Bacho, R., Loss, J.: On the adaptive security of the threshold BLS signature scheme. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 193–207. ACM Press, Los Angeles, CA, USA (Nov 7–11, 2022). <https://doi.org/10.1145/3548606.3560656>
8. Backes, M., Hanzlik, L., Klucznik, K., Schneider, J.: Signatures with flexible public key: Introducing equivalence classes for public keys. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 405–434. Springer, Heidelberg, Germany, Brisbane, Queensland, Australia (Dec 2–6, 2018). [https://doi.org/10.1007/978-3-030-03329-3\\_14](https://doi.org/10.1007/978-3-030-03329-3_14)
9. Backes, M., Hanzlik, L., Schneider-Bensch, J.: Membership privacy for fully dynamic group signatures. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2181–2198. ACM Press, London, UK (Nov 11–15, 2019). <https://doi.org/10.1145/3319535.3354257>
10. Bauer, B., Fuchsbauer, G.: Efficient signatures on randomizable ciphertexts. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 359–381. Springer, Heidelberg, Germany, Amalfi, Italy (Sep 14–16, 2020). [https://doi.org/10.1007/978-3-030-57990-6\\_18](https://doi.org/10.1007/978-3-030-57990-6_18)
11. Bauer, B., Fuchsbauer, G.: On security proofs of existing equivalence class signature schemes. Cryptology ePrint Archive, Paper 2024/183 (2024), <https://eprint.iacr.org/2024/183>
12. Bauer, B., Fuchsbauer, G., Regen, F.: On proving equivalence class signatures secure from non-interactive assumptions. In: Tang, Q., Teague, V. (eds.) Public-Key Cryptography – PKC 2024. pp. 3–36. Springer Nature Switzerland, Cham (2024)
13. Bellare, M., Crites, E.C., Komlo, C., Maller, M., Tessaro, S., Zhu, C.: Better than advertised security for non-interactive threshold signatures. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol. 13510, pp. 517–550. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–18, 2022). [https://doi.org/10.1007/978-3-031-15985-5\\_18](https://doi.org/10.1007/978-3-031-15985-5_18)
14. Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on randomizable ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 403–422. Springer, Heidelberg, Germany, Taormina, Italy (Mar 6–9, 2011). [https://doi.org/10.1007/978-3-642-19379-8\\_25](https://doi.org/10.1007/978-3-642-19379-8_25)
15. Bobolz, J., Eidens, F., Krenn, S., Ramacher, S., Samelin, K.: Issuer-hiding attribute-based credentials. In: Conti, M., Stevens, M., Krenn, S. (eds.) CANS 21. LNCS, vol. 13099, pp. 158–178. Springer, Heidelberg, Germany, Vienna, Austria (Dec 13–15, 2021). [https://doi.org/10.1007/978-3-030-92548-2\\_9](https://doi.org/10.1007/978-3-030-92548-2_9)
16. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg, Germany, Miami, FL, USA (Jan 6–8, 2003). [https://doi.org/10.1007/3-540-36288-6\\_3](https://doi.org/10.1007/3-540-36288-6_3)
17. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg, Germany, Gold Coast, Australia (Dec 9–13, 2001). [https://doi.org/10.1007/3-540-45682-1\\_30](https://doi.org/10.1007/3-540-45682-1_30)

18. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the Weil pairing. *Journal of Cryptology* **17**(4), 297–319 (Sep 2004). <https://doi.org/10.1007/s00145-004-0314-9>
19. Brandão, L., Peralta, R.: Nist first call for multi-party threshold schemes. Online (2023), <https://csrc.nist.gov/pubs/ir/8214/c/ipd>
20. Bresson, E., Stern, J., Szydło, M.: Threshold ring signatures and applications to ad-hoc groups. In: Yung, M. (ed.) *CRYPTO 2002*. LNCS, vol. 2442, pp. 465–480. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2002). [https://doi.org/10.1007/3-540-45708-9\\_30](https://doi.org/10.1007/3-540-45708-9_30)
21. Bultel, X., Lafourcade, P., Lai, R.W.F., Malavolta, G., Schröder, D., Thyagarajan, S.A.K.: Efficient invisible and unlinkable sanitizable signatures. In: Lin, D., Sako, K. (eds.) *PKC 2019, Part I*. LNCS, vol. 11442, pp. 159–189. Springer, Heidelberg, Germany, Beijing, China (Apr 14–17, 2019). [https://doi.org/10.1007/978-3-030-17253-4\\_6](https://doi.org/10.1007/978-3-030-17253-4_6)
22. Celi, S., Griffy, S., Hanzlik, L., Perez Kempner, O., Slamanig, D.: Sok: Signatures with randomizable keys. *Cryptology ePrint Archive*, Paper 2023/1524 (2023), <https://eprint.iacr.org/2023/1524> Signatures with randomizable keys. *Cryptology ePrint Archive*, Paper 2023/1524 (2023), <https://eprint.iacr.org/2023/1524>
23. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable signatures: New definitions and delegatable anonymous credentials. In: Datta, A., Fournet, C. (eds.) *CSF 2014 Computer Security Foundations Symposium*. pp. 199–213. IEEE Computer Society Press, Vienna, Austria (Jul 19–22, 2014). <https://doi.org/10.1109/CSF.2014.22>
24. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: Brickell, E.F. (ed.) *CRYPTO'92*. LNCS, vol. 740, pp. 89–105. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 1993). [https://doi.org/10.1007/3-540-48071-4\\_7](https://doi.org/10.1007/3-540-48071-4_7)
25. Chen, Y.H., Lindell, Y.: Feldman’s verifiable secret sharing for a dishonest majority. *Cryptology ePrint Archive*, Paper 2024/031 (2024), <https://eprint.iacr.org/2024/031>
26. Chen, Y.H., Lindell, Y.: Optimizing and implementing fischlin’s transform for uc-secure zero-knowledge. *Cryptology ePrint Archive*, Paper 2024/526 (2024), <https://eprint.iacr.org/2024/526>
27. Connolly, A., Deschamps, J., Lafourcade, P., Perez-Kempner, O.: Protego: Efficient, revocable and auditable anonymous credentials with applications to hyperledger fabric. In: Isobe, T., Sarkar, S. (eds.) *Progress in Cryptology - INDOCRYPT 2022 - 23rd International Conference on Cryptology in India, Kolkata, India, December 11-14, 2022, Proceedings*. Lecture Notes in Computer Science, vol. 13774, pp. 249–271. Springer (2022). [https://doi.org/10.1007/978-3-031-22912-1\\_11](https://doi.org/10.1007/978-3-031-22912-1_11)
28. Connolly, A., Lafourcade, P., Perez-Kempner, O.: Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) *PKC 2022, Part I*. LNCS, vol. 13177, pp. 409–438. Springer, Heidelberg, Germany, Virtual Event (Mar 8–11, 2022). [https://doi.org/10.1007/978-3-030-97121-2\\_15](https://doi.org/10.1007/978-3-030-97121-2_15)
29. Crites, E., Kohlweiss, M., Preneel, B., Sedaghat, M., Slamanig, D.: Threshold structure-preserving signatures. In: Guo, J., Steinfeld, R. (eds.) *Advances in Cryptology – ASIACRYPT 2023*. pp. 348–382. Springer Nature Singapore, Singapore (2023)
30. Crites, E.C., Komlo, C., Maller, M.: Fully adaptive schnorr threshold signatures. In: Handschuh, H., Lysyanskaya, A. (eds.) *CRYPTO 2023, Part I*. LNCS, vol. 14081, pp. 678–709. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2023). [https://doi.org/10.1007/978-3-031-38557-5\\_22](https://doi.org/10.1007/978-3-031-38557-5_22)

31. Crites, E.C., Komlo, C., Maller, M., Tessaro, S., Zhu, C.: Snowblind: A threshold blind signature in pairing-free groups. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part I. LNCS, vol. 14081, pp. 710–742. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2023). [https://doi.org/10.1007/978-3-031-38557-5\\_23](https://doi.org/10.1007/978-3-031-38557-5_23)
32. Crites, E.C., Lysyanskaya, A.: Delegatable anonymous credentials from mercurial signatures. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 535–555. Springer, Heidelberg, Germany, San Francisco, CA, USA (Mar 4–8, 2019). [https://doi.org/10.1007/978-3-030-12612-4\\_27](https://doi.org/10.1007/978-3-030-12612-4_27)
33. Crites, E.C., Lysyanskaya, A.: Mercurial signatures for variable-length messages. PoPETs **2021**(4), 441–463 (Oct 2021). <https://doi.org/10.2478/popets-2021-0079>
34. Das, S., Ren, L.: Adaptively secure bls threshold signatures from dddh and co-cdh. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology – CRYPTO 2024. pp. 251–284. Springer Nature Switzerland, Cham (2024)
35. Derler, D., Hanser, C., Slamanig, D.: A new approach to efficient revocable attribute-based anonymous credentials. In: Groth, J. (ed.) 15th IMA International Conference on Cryptography and Coding. LNCS, vol. 9496, pp. 57–74. Springer, Heidelberg, Germany, Oxford, UK (Dec 15–17, 2015). [https://doi.org/10.1007/978-3-319-27239-9\\_4](https://doi.org/10.1007/978-3-319-27239-9_4)
36. Derler, D., Slamanig, D.: Highly-efficient fully-anonymous dynamic group signatures. In: Kim, J., Ahn, G.J., Kim, S., Kim, Y., López, J., Kim, T. (eds.) ASIACCS 18. pp. 551–565. ACM Press, Incheon, Republic of Korea (Apr 2–6, 2018)
37. Derler, D., Slamanig, D.: Key-homomorphic signatures: definitions and applications to multiparty signatures and non-interactive zero-knowledge. Designs, Codes and Cryptography **87**(6), 1373–1413 (2019). <https://doi.org/10.1007/s10623-018-0535-9>
38. Drijvers, M., Edalatnejad, K., Ford, B., Kiltz, E., Loss, J., Neven, G., Stepanovs, I.: On the security of two-round multi-signatures. In: 2019 IEEE Symposium on Security and Privacy. pp. 1084–1101. IEEE Computer Society Press, San Francisco, CA, USA (May 19–23, 2019). <https://doi.org/10.1109/SP.2019.00050>
39. Feldman, P.: A practical scheme for non-interactive verifiable secret sharing. In: 28th FOCS. pp. 427–437. IEEE Computer Society Press, Los Angeles, CA, USA (Oct 12–14, 1987). <https://doi.org/10.1109/SFCS.1987.4>
40. Fischlin, M.: Communication-efficient non-interactive proofs of knowledge with online extractors. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 152–168. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 14–18, 2005). [https://doi.org/10.1007/11535218\\_10](https://doi.org/10.1007/11535218_10)
41. Fuchsbauer, G., Hanser, C., Kamath, C., Slamanig, D.: Practical round-optimal blind signatures in the standard model from weaker assumptions. In: Zikas, V., De Prisco, R. (eds.) SCN 16. LNCS, vol. 9841, pp. 391–408. Springer, Heidelberg, Germany, Amalfi, Italy (Aug 31 – Sep 2, 2016). [https://doi.org/10.1007/978-3-319-44618-9\\_21](https://doi.org/10.1007/978-3-319-44618-9_21)
42. Fuchsbauer, G., Hanser, C., Slamanig, D.: Practical round-optimal blind signatures in the standard model. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 233–253. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015). [https://doi.org/10.1007/978-3-662-48000-7\\_12](https://doi.org/10.1007/978-3-662-48000-7_12)
43. Fuchsbauer, G., Hanser, C., Slamanig, D.: Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. Journal of Cryptology **32**(2), 498–546 (Apr 2019). <https://doi.org/10.1007/s00145-018-9281-4>
44. Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. Journal of Cryptology **9**(3), 167–190 (Jun 1996)

45. Hanser, C., Slamanig, D.: Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 491–511. Springer, Heidelberg, Germany, Kaoshiung, Taiwan, R.O.C. (Dec 7–11, 2014). [https://doi.org/10.1007/978-3-662-45611-8\\_26](https://doi.org/10.1007/978-3-662-45611-8_26)
46. Hanzlik, L., Slamanig, D.: With a little help from my friends: Constructing practical anonymous credentials. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 2004–2023. ACM Press, Virtual Event, Republic of Korea (Nov 15–19, 2021). <https://doi.org/10.1145/3460120.3484582>
47. Katz, J.: Round optimal fully secure distributed key generation. Cryptology ePrint Archive, Paper 2023/1094 (2023), <https://eprint.iacr.org/2023/1094>
48. Labs, P.: Blast: Multilingual bls12-381 signature library. Online (2020), <https://github.com/supranational/blst>
49. Labs, P.: High performance implementation of bls12 381. Online (2021), <https://github.com/filecoin-project/blstrs>
50. Mir, O., Bauer, B., Griffy, S., Lysyanskaya, A., Slamanig, D.: Aggregate signatures with versatile randomization and issuer-hiding multi-authority anonymous credentials. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023. pp. 30–44. ACM (2023). <https://doi.org/10.1145/3576915.3623203>
51. Mitrokotsa, A., Mukherjee, S., Sedaghat, M., Slamanig, D., Tomy, J.: Threshold structure-preserving signatures: Strong and adaptive security under standard assumptions. In: Tang, Q., Teague, V. (eds.) Public-Key Cryptography – PKC 2024. pp. 163–195. Springer Nature Switzerland, Cham (2024)
52. Nanri, M.: Implementation of interactive threshold mercurial signatures. Online (2024), [https://github.com/octaviopk9/asiacrypt\\_tms](https://github.com/octaviopk9/asiacrypt_tms)
53. Nick, J., Ruffing, T., Seurin, Y.: MuSig2: Simple two-round Schnorr multi-signatures. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 189–221. Springer, Heidelberg, Germany, Virtual Event (Aug 16–20, 2021). [https://doi.org/10.1007/978-3-030-84242-0\\_8](https://doi.org/10.1007/978-3-030-84242-0_8)
54. Pedersen, T.P.: A threshold cryptosystem without a trusted party (extended abstract) (rump session). In: Davies, D.W. (ed.) EUROCRYPT’91. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg, Germany, Brighton, UK (Apr 8–11, 1991). [https://doi.org/10.1007/3-540-46416-6\\_47](https://doi.org/10.1007/3-540-46416-6_47)
55. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO’91. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 11–15, 1992). [https://doi.org/10.1007/3-540-46766-1\\_9](https://doi.org/10.1007/3-540-46766-1_9)
56. Pointcheval, D., Sanders, O.: Short randomizable signatures. In: Sako, K. (ed.) CT-RSA 2016. LNCS, vol. 9610, pp. 111–126. Springer, Heidelberg, Germany, San Francisco, CA, USA (Feb 29 – Mar 4, 2016). [https://doi.org/10.1007/978-3-319-29485-8\\_7](https://doi.org/10.1007/978-3-319-29485-8_7)
57. Pointcheval, D., Sanders, O.: Reassessing security of randomizable signatures. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 319–338. Springer, Heidelberg, Germany, San Francisco, CA, USA (Apr 16–20, 2018). [https://doi.org/10.1007/978-3-319-76953-0\\_17](https://doi.org/10.1007/978-3-319-76953-0_17)
58. Putman, C., Martin, K.M.: Selective delegation of attributes in mercurial signature credentials. In: Quaglia, E.A. (ed.) Cryptography and Coding. pp. 181–196. Springer Nature Switzerland, Cham (2024) signature credentials. In: Quaglia, E.A.

- (ed.) *Cryptography and Coding*. pp. 181–196. Springer Nature Switzerland, Cham (2024)
59. Sanders, O., Traoré, J.: Compact issuer-hiding authentication, application to anonymous credential. *Proc. Priv. Enhancing Technol.* **2024**(3), 645–658 (2024). <https://doi.org/10.56553/POPETS-2024-0097>
  60. Tessaro, S., Zhu, C.: Short pairing-free blind signatures with exponential security. In: Dunkelman, O., Dziembowski, S. (eds.) *EUROCRYPT 2022, Part II*. LNCS, vol. 13276, pp. 782–811. Springer, Heidelberg, Germany, Trondheim, Norway (May 30 – Jun 3, 2022). [https://doi.org/10.1007/978-3-031-07085-3\\_27](https://doi.org/10.1007/978-3-031-07085-3_27)





# HARTS: High-Threshold, Adaptively Secure, and Robust Threshold Schnorr Signatures

Renas Bacho<sup>1,3</sup>, Julian Loss<sup>1</sup>, Gilad Stern<sup>2</sup>, and Benedikt Wagner<sup>4</sup>

<sup>1</sup> CISPA Helmholtz Center for Information Security, Saarbrücken, Germany  
`{renas.bacho,loss}@cispa.de`

<sup>2</sup> Tel Aviv University, Tel Aviv-Yafo, Israel  
`giladstern@tauex.tau.ac.il`

<sup>3</sup> Saarland University, Saarbrücken, Germany

<sup>4</sup> Ethereum Foundation, Berlin, Germany  
`benedikt.wagner@ethereum.org`

**Abstract.** Threshold variants of the Schnorr signature scheme have recently been at the center of attention due to their applications to cryptocurrencies. However, existing constructions for threshold Schnorr signatures among a set of  $n$  parties with corruption threshold  $t_c$  suffer from at least one of the following drawbacks: (i) security only against static (i.e., non-adaptive) adversaries, (ii) cubic or higher communication cost to generate a single signature, (iii) strong synchrony assumptions on the network, or (iv)  $t_c + 1$  are sufficient to generate a signature, i.e., the corruption threshold of the scheme equals its reconstruction threshold. Especially (iv) turns out to be a severe limitation for many asynchronous real-world applications where  $t_c < n/3$  is necessary to maintain liveness, but a higher signing threshold of  $n - t_c$  is needed. A recent scheme, ROAST, proposed by Ruffing et al. (ACM CCS '22) addresses (iii) and (iv), but still falls short of obtaining subcubic complexity and adaptive security.

In this work, we present HARTS, the *first* threshold Schnorr signature scheme to incorporate all these desiderata. More concretely:

- HARTS is adaptively secure and remains fully secure and operational even under asynchronous network conditions in the presence of up to  $t_c < n/3$  malicious parties. This is optimal.
- HARTS outputs a Schnorr signature of size  $\lambda$  with a near-optimal amortized communication cost of  $O(\lambda n^2 \log n)$  bits and a single online round per signature.
- HARTS is a *high-threshold* scheme: no fewer than  $t_r + 1$  signature shares can be combined to yield a full signature, where any  $t_r \in [t_c, n - t_c]$  is supported. This especially covers the case  $t_r \geq 2n/3 > 2t_c$ . This is optimal.

We prove our result in a modular fashion in the algebraic group model. At the core of our construction, we design a new simple and adaptively secure high-threshold asynchronous verifiable secret sharing (AVSS) scheme which may be of independent interest.

---

This work was done while the author was at CISPA Helmholtz Center for Information Security.

**Keywords:** Threshold Signatures · Schnorr Signatures · Adaptive Security · Robustness · High-Threshold · Asynchronous Network

## 1 Introduction

A *threshold signature* [45, 46] scheme is a special type of digital signature scheme that allows any set of  $t_r + 1$  signers in a system of  $n$  parties to jointly generate a compact signature  $\sigma$  on a message  $m$ . On the other hand, this should be infeasible for  $t_r$  or less signers. Over the last two decades, many threshold versions of the Schnorr signature scheme [74] have been proposed [23, 32, 38, 55, 65, 68, 73, 80–82]. Collectively, these schemes offer a great variety of trade-offs between efficiency and robustness to adverse signer behaviour and network conditions. A recent work, ROAST [73], combines several of these desirable features into a single scheme which supports high reconstruction threshold (see below) and maintains liveness even under full asynchrony. Unfortunately, the scheme is rather inefficient: creating a single signature costs  $O(\lambda n^3 + n^4)$  bits and  $O(n)$  rounds of communication, where  $\lambda$  denotes the size of a signature.

**Our Contribution.** In this work, we propose HARTS, a novel threshold Schnorr signature scheme that improves significantly over prior works. Concretely, HARTS has the following properties:

- **Adaptive Security:** HARTS is secure against *strongly adaptive corruptions*.
- **Asynchrony:** HARTS remains fully secure and operational against up to (optimal)  $t_c < n/3$  corrupted parties in a *fully asynchronous* network where message delivery between honest parties can take longer than expected [66].
- **Efficiency:** HARTS allows for message-independent, offline generation of (ephemeral) nonces with an amortized communication cost of  $O(\lambda n^2 \log n)$  bits per nonce and  $O(1)$  round complexity. Upon a signing request, it produces a Schnorr signature in a single round with a total communication cost of  $O(\lambda n^2)$  bits, where each party only sends a single field element.
- **High-threshold:** HARTS is a *high-threshold* signature scheme satisfying the following features: (i) a signing session results in a valid signature even in the presence of up to  $t_c$  malicious parties that try to prevent the other parties from generating a signature, (ii) a signature cannot be created given less than  $t_r + 1$  signature shares, where any  $t_r \in [t_c, n - t_c)$  is supported. In particular, this also covers the case  $t_r \geq 2n/3 > 2t_c$  and thus offers more flexibility. This notion of enhanced security has found many applications and real-world significance in recent years, especially in the context of consensus and blockchain systems [12, 52, 83] where signatures from a Byzantine quorum of size  $n - t_c \geq 2t_c + 1 > t_c + 1$  are needed.

We refer to Table 1 for a complete overview and comparison of our scheme’s properties with existing schemes from the literature. A caveat is that we need to

**Table 1.** Comparison table of some relevant threshold Schnorr signatures.

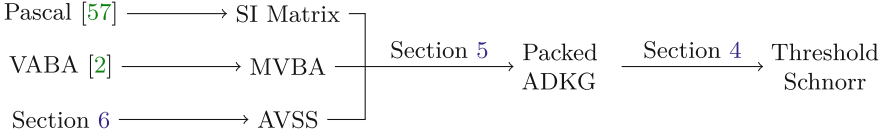
Scheme	Robust	Corrupt	Reconst	Adapt	Commun	Rounds
FROST [65]	✗	$t_c < n$	$t_r = t_c$	✗	$O(\lambda n^2)$	2
Sparkle [38]	✗	$t_c < n$	$t_r = t_c$	✓	$O(\lambda n^2)$	3
GJKR [55]	(✓)	$t_c < n/2$	$t_r = t_c$	✗	$O(\lambda n^3)$	3 BC
ROAST [73]	✓	$t_c < n^\dagger$	$t_r < n - t_c$	✗	$O(\lambda n^3 + n^4)$	$O(n)$
SPRINT [18]	✓	$t_c < n/3$	$t_r = t_c$	✗	$O(\lambda n^2)$	3 BC
GS23 [57]	✓	$t_c < n/3$	$t_r = t_c$	✗	$O(\lambda n)$	$O(1)$
<b>Our work</b>	✓	$t_c < n/3$	$t_r < n - t_c$	✓	$O(\lambda n^2)$	$O(1)$

**Robust** denotes robustness, i.e., signing sessions always produce valid signatures. The work [55] assumes a synchronous network and thus is not robust in our sense. The protocols [38, 65] are not robust in synchrony, as a malicious signer in the signing set can cause the session to abort or the final signature to be invalid, and do not necessarily terminate in full asynchrony (we refer to [73] for a discussion on that). However, we note that both protocols still remain unforgeable in asynchrony. **Corrupt** denotes corruption threshold.  $\dagger$  ROAST has a weaker notion of robustness and does not guarantee signature generation for  $t_c \geq n/3$ . **Reconst** denotes reconstruction threshold. **Adapt** denotes adaptive security. **Commun** denotes (amortized) communication cost per signature in bits. **Rounds** denotes number of rounds per signature. The works [18, 55] assume an (atomic) broadcast channel BC that parties can access: the former employs a blockchain for this, while the latter does not specify how to implement it. The work [57] and ours can generate many batches of  $O(n)$  ephemeral nonces in a message-independent offline phase with round complexity  $O(1)$  per batch and a single message-dependent online round per signature upon a signing request.

make minimal use of secure erasures for our security proof. The reason is that without secure erasures, we would need to *explain* simulated zero-knowledge arguments by revealing appropriate random coins. We leave it as an interesting problem for future work to analyze if the arguments we employ are explainable (cf. Remark 6 for more details). In that case, our protocol would work without assuming erasures. We emphasize, however, that secure erasures do not trivialize the task of achieving adaptive security at all [27, 34, 35, 49].

**A Modular Approach.** We build HARTS by following a modular approach, which is summarized in Fig. 1 and outlined in more detail below.

- **Threshold Schnorr Signatures from ADKG.** Building upon the technique of Gennaro et al. [54], we construct a generic high-threshold and robust threshold Schnorr signature scheme. As a building block, we use a (packed) high-threshold asynchronous distributed key generation (ADKG) protocol. We prove unforgeability of this scheme against an adaptive adversary in the algebraic group model (AGM) [50] based on the security of the (packed) ADKG protocol and the one-more discrete logarithm assumption.
- **Packed ADKG from AVSS.** We give a generic construction (cf. Fig. 2) for an efficient packed high-threshold ADKG from a high-threshold asynchronous



**Fig. 1.** Overview of our framework to construct high-threshold and robust threshold Schnorr signatures.

verifiable secret sharing (AVSS) scheme using the technique of superinvertible matrices [60]. We prove the adaptive security of this construction by reduction to the security of the AVSS scheme and the underlying consensus primitives.

- **New High-Threshold AVSS.** We design a simple high-threshold AVSS scheme and give an adaptive security proof. This gives the first pairing-free, adaptively secure AVSS scheme with quadratic communication cost (cf. Table 2 for a comparison with existing schemes). With our new AVSS scheme and building blocks from the literature, we instantiate our framework, yielding a threshold Schnorr signature scheme with (amortized) communication cost of  $O(\lambda n^2 \log n)$  bits per signature.

### 1.1 Technical Overview

In the following, we provide a technical overview of our work.

**Starting Point: Robust Threshold Schnorr Signatures.** Our starting point is the construction for robust threshold Schnorr signatures by Gennaro et al. [54]. First, we recall the (single-party) Schnorr signature scheme. For this, let  $\mathbb{G} = \langle g \rangle$  be a group of prime order  $p$  with generator  $g$ . The secret key is a random element  $\text{sk} \leftarrow \mathbb{Z}_p$  and the public key is  $\text{pk} := g^{\text{sk}}$ . To sign a message  $m$ , the party samples a random element  $r \leftarrow \mathbb{Z}_p$  and computes the signature on  $m$  as  $\sigma := (R, s)$  where  $s = \text{H}(\text{pk}, R, m) \cdot \text{sk} + r \in \mathbb{Z}_p$  and  $R = g^r$ . Here,  $\text{H}: \{0, 1\}^* \rightarrow \mathbb{Z}_p$  is a hash function (modeled as a random oracle). Verification of the signature  $(R, s)$  is done by checking  $g^s = R \cdot \text{pk}^c$  where  $c := \text{H}(\text{pk}, R, m)$ . Now let us switch to the multi-party setting in which  $n$  parties  $P_1, \dots, P_n$  want to jointly create a signature  $\sigma$  on  $m$ . For convenience we assume that parties have already established a  $(t_r, n)$ -threshold key setup, e.g., by running a distributed key generation (DKG) protocol. Concretely, this means that each party  $P_i$  has a share  $\text{sk}_i$  of the secret key  $\text{sk}$  such that any set of  $t_r + 1$  shares uniquely determine the secret key and the public key shares  $\text{pk}_i := g^{\text{sk}_i}$  are known to all parties. In order to transform the Schnorr signature scheme into a  $(t_r, n)$ -threshold signature scheme, Gennaro et al.’s insight was to run a DKG protocol to generate shares  $r_i$  of a secret nonce  $r$  for parties along with associated public shares  $R_i = g^{r_i}$  and  $R = g^r$ . To sign a message  $m$ , each party  $P_i$  computes its share of the signature on  $m$  as  $\sigma_i := (R_i, s_i)$  where  $s_i = \text{H}(\text{pk}, R, m) \cdot \text{sk}_i + r_i$ . Verification of a signature share  $\sigma_i$  is done with respect to the public key share  $\text{pk}_i$  and the public nonce share  $R_i$ . It can be seen that any  $t_r + 1$  valid signature shares recover the full signature  $\sigma = (R, s)$ . However, a major drawback of this approach is its efficiency:

parties need to run a DKG protocol each time they want to sign a new message. Using a state-of-the-art asynchronous DKG protocol in terms of efficiency [2, 42], this yields a communication cost of  $O(\lambda n^3)$  bits per signature. On the other hand, assuming nonce shares have already been generated, each party can locally compute its signature share and send it to all other parties, which costs only  $O(\lambda n^2)$  bits of communication and a single asynchronous round.

**Regaining Efficiency: Superinvertible Matrices.** Introduced by Hirt and Nielsen [60] in the context of multi-party computation (MPC) protocols, we use the technique of superinvertible (SI) matrices to construct an  $(\ell, t_c, t_r, n)$ -packed ADKG protocol for more efficient multi-nonce generation. Informally, such a protocol executed by  $n$  parties has the following property in the presence of up to  $t_c$  malicious parties: it outputs  $\ell$  independent keys distributed among the parties such that each of them can be reconstructed independently from the others with reconstruction threshold  $t_r$ . Our construction has the following parameters: it generates  $\ell = t_c + 1$  keys with (arbitrary) reconstruction threshold  $t_r < n - t_c$  in the presence of  $t_c < n/3$  malicious parties. Our construction (cf. Fig. 2) follows the usual flow of an ADKG protocol with some tweaks in the parameters in order to apply an SI matrix at the end. We briefly elaborate on this. Each party  $P_i$  samples a random element  $s_i \leftarrow \mathbb{Z}_p$  and shares it via an  $(t_c, t_r, n)$ -threshold asynchronous verifiable secret sharing (AVSS) scheme where  $s_i$  lies on some polynomial  $f_i \in \mathbb{Z}_p[X]$  of degree  $t_r$ . Then, parties agree on a set  $I \subset [n]$  of  $n - t_c$  dealers whose AVSS sharings completed successfully using a consensus tool. Say  $I = \{1, \dots, n - t_c\}$  so that after this phase, each party  $P_i$  has shares  $f_1(i), \dots, f_{n-t_c}(i)$ . Instead of just summing up these shares, as is done usually in an ADKG, parties take different linear combinations given by the rows of an  $(\ell, n - t_c)$ -dimensional SI matrix  $\text{SI}$  to obtain  $\ell$  new shares  $r_1(i), \dots, r_\ell(i)$ . The describing property of an SI matrix now tells us that if at least  $\ell$  input secrets are independent and uniformly random, then the  $\ell$  output secrets are also guaranteed to be independent and uniformly random. Since there are at most  $t_c$  malicious parties, we know that  $|I| - t_c \geq t_c + 1$  of the dealers specified by the set  $I$  are honest. Thus, we can set  $\ell := t_c + 1$ . Similar constructions were recently introduced in [18, 57, 77] for the same purpose of efficient multi-nonce generation. These constructions, however, employ low-threshold AVSS schemes with  $t_r = t_c < n/3$ . To the best of our knowledge, the only existing high-threshold AVSS schemes are [2, 5, 44, 64], each of them with its own limitations. Kokoris-Kogias et al.’s AVSS [64] has cubic communication cost, resulting in prohibitive  $\Omega(\lambda n^4)$  communication to share  $t_c + 1$  nonces. Alhaddad et al. [5] provide a generic construction for AVSS with quadratic communication cost, but lacking a proof of adaptive security. Das et al.’s AVSS [44] is based on publicly verifiable secret sharing (PVSS) and has quadratic communication cost, but also lacking a proof of adaptive security<sup>1</sup>. Abraham et al.’s AVSS [2] relies on the KZG polynomial commitment scheme [62] that requires pairings (and trusted setup) which is not suitable for Schnorr signatures.

<sup>1</sup> Crucially, no adaptively secure PVSS for field elements is known to date.

**HAVSS: New AVSS Scheme to the Rescue.** We take insights from both protocols, Bingo [2] and HAVEN [5], and combine certain aspects to obtain a simple high-threshold AVSS scheme called HAVSS. On a high level, our AVSS scheme works as follows. The designated dealer  $P_d$  holds a secret  $s \in \mathbb{Z}_p$  as input that it wants to share among all parties. For this, it samples a bivariate polynomial  $S \in \mathbb{Z}_p[X, Y]$  of degree  $t_r$  in  $X$  and  $t_c$  in  $Y$  such that  $S(0, 0) = s$ . The goal is to let each party  $P_i$  receive the column polynomial  $C_i(Y) := S(i, Y)$  assigned to it so that it can recover its share  $s_i := S(i, 0) \in \mathbb{Z}_p$  of the secret  $s$ . Note that the shares  $s_i$  lie on a polynomial  $S(X, 0) \in \mathbb{Z}_p[X]$  of degree  $t_r$ . We follow a simple two-step approach which results in an  $(n \times n)$ -dimensional matrix whose entry at coordinates  $a, b \in [n]$  is  $S(a, b)$ . First, the dealer reliably broadcasts Pedersen commitments  $\{\text{com}_1, \dots, \text{com}_{t_r+1}\}$  on the column polynomials  $C_1(Y), \dots, C_{t_r+1}(Y)$ , from which parties can locally (by interpolation) derive the commitments  $\{\text{com}_1, \dots, \text{com}_n\}$  to all  $n$  column polynomials  $C_1(Y), \dots, C_n(Y)$ . Following this,  $P_d$  sends each party  $P_i$  shares  $\{C_1(i), C_2(i), \dots, C_n(i)\}$  on each other party's assigned column polynomial, along with proofs that the openings are correct. This can be thought of as sending to  $P_i$  the evaluations along the row  $R_i(X) := S(X, i)$ . Whenever a party  $P_i$  receives a row with correct opening proofs, it sends every other party  $P_j$  the share  $C_j(i)$  (along with the proof sent by the dealer) on its column polynomial  $C_j(Y)$ . In this way, it is guaranteed that each party  $P_i$  obtains at least  $t_c + 1$  shares on its column polynomial  $C_i(Y)$  and can recover its share  $s_i = C_i(0) = S(i, 0)$  of the dealer's initial secret  $s$ . To guarantee unanimous termination, we employ a Bracha-style termination gadget [22] in which parties send their approval to all parties upon receiving a correct row, and echo other parties' approvals upon seeing a total of  $n - t_c$  approvals. HAVSS has a near-optimal communication cost of  $O(\lambda n^2 \log n)$  per sharing (cf. Table 2). In combination with the aforementioned technique of superinvertible matrices, we are able to construct a packed ADKG protocol that outputs  $\ell = t_c + 1 \in O(n)$  nonces with  $O(\lambda n^3 \log n)$  bits and  $O(1)$  rounds of communication. As a result, we achieve an amortized communication cost of  $O(\lambda n^2 \log n)$  per generated nonce and Schnorr signature.

**Handling Adaptive Corruptions.** To prove adaptive security, our starting point is the recent work of Bacho and Loss [8] who introduced a new security notion for DKG protocols called *oracle-aided simulatability*. Loosely speaking, this notion states the existence of an efficient simulator  $\text{Sim}$  that on input  $k$  group elements  $\xi_1, \dots, \xi_k \in \mathbb{G}$  can simulate an execution of the DKG protocol under adaptive corruptions while having  $(k - 1)$ -time access to a discrete logarithm oracle  $\text{DL}_{\mathbb{G}, g}$ . With this notion of security for DKG, they show a reduction from the one-more discrete logarithm (OMDL) assumption [15] of degree  $k$  to the unforgeability of the threshold BLS signature scheme against an adaptive adversary. Their reduction internally runs  $\text{Sim}$  (on input the OMDL challenge  $\xi_1, \dots, \xi_k \in \mathbb{G}$ ) in order to simulate an execution of the DKG protocol as part of the broader simulation of the unforgeability experiment. To emulate the oracle  $\text{DL}_{\mathbb{G}, g}$  for the simulator  $\text{Sim}$ , the reduction simply forwards any query  $\text{Sim}$  makes to its own oracle. We want to employ a similar strategy to simulate the

**Table 2.** Comparison table of some relevant AVSS schemes.

Scheme	Adaptive	High-Threshold	Pairing-Free	No Trust	Commun
Backes et al. [11]	✗	✗	✗	✗	$O(\lambda n^2)$
hbACSS [84]	✗	✗	✓	✓	$O(\lambda n^2 \log n)$
GoAVSS [78]	✗	✗	✓	✓	$O(\lambda n^2)$
Cachin et al. [25]	✗	(✓)*	✓	✓	$O(\lambda n^3)$
Das et al. [44]	✗	✓	✓	✓	$O(\lambda n^2)$
HAVEN [5]	✗	✓	✓	✓	$O(\lambda n^2 \log n)$
HAVEN++ [4] (concurrent)	✗	✓	✓	✓	$O(\lambda n^2 \log n)$
Bingo [2]	✓	✓	✗	✗	$O(\lambda n^2)$
Kokoris et al. [64]	✓	✓	✓	✓	$O(\lambda n^3)$
<b>HAVSS (our work)</b>	✓	✓	✓	✓	$O(\lambda n^2 \log n)$

**Adaptive** denotes adaptive security. **High-Threshold** denotes support for high reconstruction threshold. \*The work [25] only achieves suboptimal corruption threshold  $t_c < n/4$  (in asynchrony). **Pairing-Free** denotes suitability for pairing-free groups. **No Trust** denotes no trusted setup other than a uniform reference string (URS). The works [2, 11] rely on the structured powers-of-tau setup [72]. **Commun** denotes communication cost per sharing in bits.

executions for multi-nonce and key generation. However, we encounter several challenges when trying to adopt this strategy naively. For the remainder of this overview, we assume for simplicity that parties employ a regular single-output ADKG protocol for nonce generation instead of a packed one.

**Challenges in Our Context.** Very recently, Crites et al. [38] gave an adaptive security proof for their threshold Schnorr signature scheme under the algebraic OMDL assumption. This assumption is widely used in the context of multi-party Schnorr signatures [16, 70]. So far, the scheme by Crites et al. is the only threshold Schnorr signature with adaptive security. In their proof, corruption queries are simulated using the oracle  $DL_{\mathbb{G},g}$  and signing queries are simulated using honest-verifier zero-knowledge and by programming the random oracle suitably. Omitting details, their simulator essentially samples random signature shares  $\sigma_i \leftarrow_s \mathbb{Z}_p$  for honest parties and retroactively defines the public nonce shares  $R_i$  by suitably programming the random oracle. To make this strategy work in our context, the (packed) ADKG protocol NDKG for nonce generation would have to be *fully secret* in the sense of Gennaro et al. [54], i.e., there exists an efficient simulator that on input a group element  $R \in \mathbb{G}$  can simulate an execution of NDKG that terminates with  $R$  as public nonce. While there are constructions [1, 61] relying on the single-inconsistent player (SIP) technique that achieve this kind of property, these fail to work in our setting. On a high level, the SIP technique allows to reveal the internal state of all but one single party which is chosen randomly at the beginning of the protocol execution. For this to work, the sharing of the single-inconsistent party has to be included in the final transcript of the DKG. However, in an asynchronous DKG, there is only the guarantee that the sharings of at most  $n - t_c$  parties are included<sup>2</sup>, of

<sup>2</sup> This feature is specific to the asynchronous network model and necessary to maintain liveness of protocols. For more details, we refer to [3, 44].

which  $t_c$  may be corrupted. Therefore, the probability that the sharing of the single-inconsistent party is included is bounded by  $(n - 2t_c)/(n - t_c) \leq 1/2$ . Multiple applications of an ADKG would therefore lead to a negligible success probability of the reduction. Therefore, to deal with adaptive corruptions in our setting, a new approach is required.

**Combining Different Proof Strategies.** In their work, Bellare et al. [16] provide a security reduction from the OMDL assumption to the security of the FROST1 and FROST2 schemes under a static adversary. Their reduction uses the discrete logarithm oracle  $\text{DL}_{\mathbb{G},g}$  to answer certain signing queries. Essentially, the values for the nonce  $R_j$ , the challenge  $c_j$ , and the public key shares  $\text{pk}_i = g^{\text{sk}_i}$  are fixed and determine the signature shares  $\sigma_{j,i}$  by the relation  $g^{\sigma_{j,i}} = R_j \cdot \text{pk}_i^{c_j}$ . We observe that a similar strategy could be useful for our scheme, in particular in combination with previously explained oracle-aided simulatability. And indeed, combining these two proof strategies [8, 16] (almost!) succeeds: using oracle-aided simulators to simulate executions of IDKG and NDKG along with corruption queries<sup>3</sup>, and at the same time using the oracle  $\text{DL}_{\mathbb{G},g}$  separately to answer signing queries. However, trying to employ this approach as it currently stands, we exceed the number of allowed queries to the oracle  $\text{DL}_{\mathbb{G},g}$  prescribed by the OMDL challenge: assume the reduction simulating the unforgeability game uses  $\text{DL}_{\mathbb{G},g}$  on input  $g^{\sigma_{j,i}}$  to answer a signing query for party  $P_i$  and nonce  $R_j$ . If  $P_i$  gets corrupted later on, the simulators for IDKG and the  $j$ -th execution of NDKG that generated  $R_j$  might make discrete logarithm queries such that they can internally compute the secret key share  $\text{sk}_i$  and the secret nonce share  $r_{j,i}$ , respectively. Three discrete logarithm oracle queries have been made to return the values  $\sigma_{j,i}, \text{sk}_i, r_{j,i}$ , although by the identity  $\sigma_{j,i} = c_j \cdot \text{sk}_i + r_{j,i}$  two queries would suffice. To resolve this issue we have to (i) adapt the original definition of oracle-aided simulatability delicately and (ii) cleverly design the reduction to limit the number of its queries to  $\text{DL}_{\mathbb{G},g}$ .

## 1.2 More on Related Work

We discuss related work on threshold signatures and DKG, and give a brief overview on other asynchronous threshold Schnorr signatures. In the full version, we discuss related work on AVSS and further threshold Schnorr signatures with a focus on robustness, high-threshold, and efficiency. Additionally, in Sect. 6.3, we provide a detailed comparison to previous (and concurrent) AVSS schemes relevant to our work.

**Threshold Signatures.** Most of the threshold signature schemes [14, 39, 63, 69] focus on threshold DSA/ECDSA and threshold Schnorr [28, 37, 38, 40, 53, 65], mainly due to their significance in blockchain systems and cryptocurrency wallets. Among the threshold Schnorr signatures, only the work [38] provides adaptive security. Further, several protocols for threshold RSA signatures were proposed [6, 63, 76] from which only [6] provides adaptive security. In the domain

<sup>3</sup> Here, IDKG denotes the initial ADKG protocol employed for key generation.



of pairing-based threshold signatures, there are several constructions [21, 36, 41, 67] from which [21, 41, 67] provide adaptive security. A recent work [10] studies rewinding-free, adaptively secure threshold signatures without pairings. For a comprehensive survey on threshold signatures, we further refer to [75].

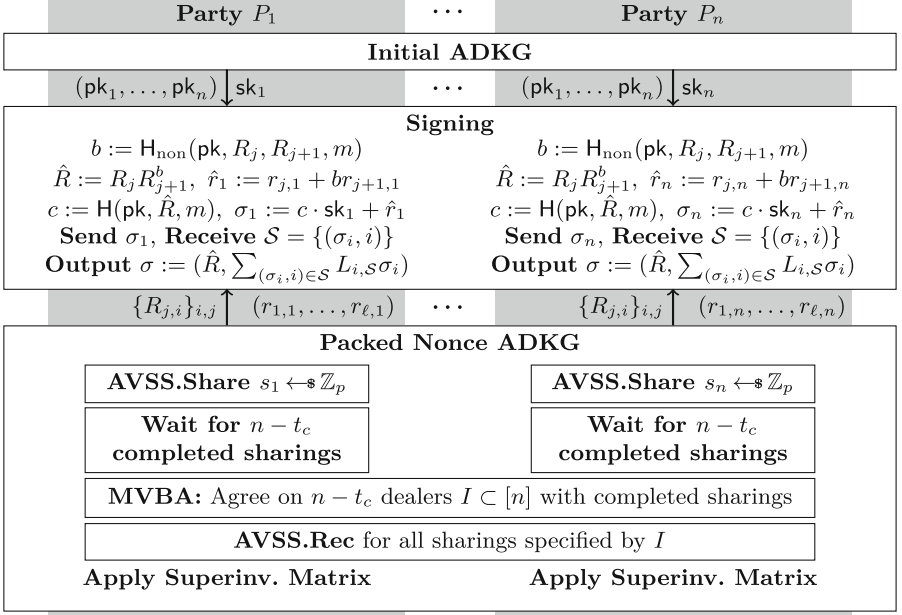
**Asynchronous Threshold Schnorr Signatures.** In recent years, several constructions for robust threshold Schnorr signatures have been proposed [18, 57, 73]. Closest to our work are the constructions in [18, 57], which follow the same high-level idea of running an ADKG protocol in combination with a superinvertible matrix for nonce generation. However, these protocols only provide low-threshold reconstruction with  $t_r = t_c < n/3$ . Essentially, the reason for this is that both rely on a low-threshold AVSS, which cannot be used in the high-threshold setting without sacrificing security. Further, the protocol in [77] uses online error correction [31] which inherently requires  $t_r < n/3$ . For security reasons, the protocol in [18] only allows parties to sign batches of  $t_c + 1$  messages (and not individual messages). On the other hand, ROAST [73] supports high-threshold reconstruction and works fundamentally different. Essentially, it transforms FROST [65] into a protocol for robust and asynchronous threshold signatures by running  $n - t_c + 1$  concurrent signing sessions of FROST in such a clever way that guarantees successful termination of at least one of these sessions. In particular, it inherits the high-threshold property of FROST. Finally, we stress that none of the works in this category achieve adaptive security.

**Distributed Key Generation.** Most of the DKG protocols assume an underlying synchronous network [7, 29, 55, 58, 61, 65, 79]. Among these protocols, only the ones in [7, 29, 61] provide adaptive security. On the other hand, DKGs in the asynchronous setting have only recently attracted attention [2, 3, 42, 44, 64]. Among these, only the works of Abraham et al. [2] and Kokoris-Kogias et al. [64] provide adaptive security. The protocols in [2, 42, 64] provide high-threshold reconstruction of the key with optimal resilience threshold, but we note that [64] is substantially less efficient than the other two protocols [2, 42]. The asynchronous DKG protocols have cubic communication cost except the one in [64] which has quartic communication cost.

**Concurrent Work.** Concurrent with our work, another work on AVSS has appeared [4]. While their construction is very similar to ours and has the same properties as HAVEN (cf. Table 2), the authors do not consider adaptive security and especially not in the context of adaptive security for asynchronous DKG protocols or threshold Schnorr signatures.

### 1.3 Outline of the Paper

The paper is organized as follows. In Sect. 2, we define relevant preliminaries. In Sect. 3, we define the model of syntax and security of a robust threshold signature scheme relevant for this work. In Sect. 4, we give a generic construction for a high-threshold, robust, and efficient threshold Schnorr signature scheme and prove it adaptively secure in the AGM. In Sect. 5, we give a generic construction



**Fig. 2.** Overview of our protocol to generate threshold Schnorr signatures.

for an efficient packed ADKG protocol and prove it adaptively secure from its building blocks. In Sect. 6, we present our new high-threshold AVSS scheme and prove it adaptively secure. In Sect. 7, we instantiate our framework to obtain HARTS, and evaluate the communication and round complexity of it. In the full version, we discuss further related work on AVSS schemes and threshold Schnorr signatures. Also, we cover there additional preliminaries and definitions relevant for the paper. Due to space constraints, we defer security proofs and additional figures to the full version.

## 2 Preliminaries and Model

In this section, we fix notation and preliminaries for our paper.

**General Notation.** Let  $\lambda$  denote the security parameter. Throughout the paper, we assume that global parameters  $par := (\mathbb{G}, p, g)$  implicitly parameterized by  $\lambda$  are fixed and known to all parties. Here,  $\mathbb{G}$  is a cyclic group of prime order  $p$  generated by  $g$ . For two integers  $a \leq b$ , we define the set  $[a, b] := \{a, \dots, b\}$ ; if  $a = 1$ , we denote this set by  $[b]$ , and if  $a = 0$ , we denote it by  $\llbracket b \rrbracket$ . For an element  $x$  in a finite set  $S$ , we write  $x \leftarrow_{\$} S$  to denote that  $x$  was sampled from  $S$  uniformly at random. All our algorithms may be randomized, unless stated otherwise. We use the acronym PPT to mean probabilistic polynomial-time. By  $x \leftarrow A(x_1, \dots, x_n)$  we denote running algorithm  $A$  on inputs  $(x_1, \dots, x_n)$  and uniformly random coins and then assigning its output to  $x$ .

**Adversarial and Network Model.** We consider a complete network of  $n$  parties  $P_1, \dots, P_n$  (modeled as PPT machines) connected by bilateral private and authenticated channels<sup>4</sup>. We consider an *asynchronous* network model, i.e., any message can be delayed arbitrarily under the constraint that messages sent between correct parties must eventually be delivered. We consider an adversary who can corrupt up to  $t_c < n/3$  parties maliciously and may cause them to deviate from the protocol arbitrarily. We refer to  $t_c$  as the corruption threshold and to  $t_r \in [t_c, n - t_c)$  as the reconstruction threshold. Further, the adversary is *strongly adaptive* and can choose its corruptions at any time during the protocol execution. When it corrupts a party, it can delete or substitute any undelivered messages that this party sent while being correct. We refer to the correct parties as *honest* and to the malicious parties as *corrupt*.

**Public Key Infrastructure.** As common in this line of work on distributed cryptographic protocols [2, 18], we assume that parties have established a bulletin board public key infrastructure (PKI) before the protocol execution. Concretely, this means that every party  $P_i$  has a verification-signing key pair  $(vk_i, sik_i)$  for a digital signature scheme, where  $vk_i$  is known to all parties but  $sik_i$  is known only to  $P_i$ . For this, we assume that each party generates its keys locally (where corrupt parties may choose their keys arbitrarily) and then makes its verification key known to everybody using a public bulletin board. These keys are used to provide authentication. In particular, we assume that parties sign each message before they send it to other parties.

**Algebraic Group Model.** In the algebraic group model (AGM) [50], all algorithms are treated as algebraic: whenever an algorithm outputs a group element, it must also provide a representation of that element with respect to all of the inputs the algorithm has received so far. Formally, an algorithm  $A$  is called *algebraic* (over a group  $\mathbb{G}$ ) if for all group elements  $h \in \mathbb{G}$  that  $A$  outputs, it additionally outputs a vector  $\mathbf{z}_\zeta = (z_1, \dots, z_m)$  of integers such that  $h = \prod_{i \in [m]} g_i^{z_i}$ , where  $\zeta = (g_1, \dots, g_m) \in \mathbb{G}^m$  is the list of group elements  $A$  has received so far.

**Computational Assumptions.** We rely on the standard *one-more discrete logarithm (OMDL) assumption* [15] for our security proofs. Throughout the paper, we denote by  $DL_{\mathbb{G},g}$  an oracle that on input  $\xi := g^z \in \mathbb{G}$  returns the discrete logarithm  $z \in \mathbb{Z}_p$  of  $\xi$  to base  $g$ .

**Definition 1 (OMDL Assumption).** *Let  $\mathbb{G}$  be a cyclic group of prime order  $p$  generated by  $g$  and  $DL_{\mathbb{G},g}$  as defined above. For an algorithm  $A$  and  $k \in \mathbb{N}$ , we consider the following experiment:*

- Offline Phase. *Sample  $(z_1, \dots, z_k) \leftarrow \mathbb{Z}_p^k$  and set  $\xi_i := g^{z_i} \in \mathbb{G}$  for all  $i \in [k]$ .*
- Online Phase. *Run  $A$  on input  $(\mathbb{G}, p, g)$  and  $(\xi_1, \dots, \xi_k)$ . Here,  $A$  gets access to the oracle  $DL_{\mathbb{G},g}$ .*

---

<sup>4</sup> When implementing those channels, one has to make sure that they are secure in the presence of adaptive corruptions. For efficient implementations of these, we refer to the early works [13, 71].

- **Winning Condition.** Let  $(z'_1, \dots, z'_k)$  denote the output of  $A$ . Return 1 if (i)  $z'_i = z_i$  for all  $i \in [k]$ , and (ii)  $\text{DL}_{\mathbb{G},g}$  was queried at most  $k - 1$  times during the online phase. Otherwise, return 0.

We say that the one-more discrete logarithm assumption of degree  $k$  holds relative to  $(\mathbb{G}, p, g)$  if for any PPT algorithm  $A$ , the probability that the above experiment outputs 1 is negligible in  $\lambda$ . Further, the discrete logarithm assumption (DLOG) is the one-more discrete logarithm assumption of degree  $k = 1$ .

## 2.1 Cryptographic and Consensus Primitives

In this section, we formally define syntax and security notions of the cryptographic and consensus primitives used in the paper.

**Multivalued Validated Byzantine Agreement.** A *multivalued validated Byzantine agreement (MVBA) protocol* [26] allows a set of parties, each holding an input  $v_i \in V$  from a value set  $V$  with  $|V| \geq 2$ , to agree on a common output value  $v' \in V$  satisfying a predefined external validity function  $\text{Val}: V \rightarrow \{0, 1\}$ . A value  $v \in V$  is said to be *externally valid* if  $\text{Val}(v) = 1$ . We formally define an MVBA protocol as follows.

**Definition 2 (MVBA Protocol).** Let  $\Pi$  be a protocol executed by  $n$  parties  $P_1, \dots, P_n$ , where each party  $P_i$  holds  $v_i \in V$  as input, and let  $\text{Val}: V \rightarrow \{0, 1\}$  be an external validity function. We say that  $\Pi$  is a  $(t_c, n)$ -secure MVBA protocol if whenever at most  $t_c$  parties are corrupted the following properties hold. (i) **External Validity:** If every honest party's input is externally valid, then every honest party  $P_i$  that outputs a value outputs an externally valid value  $v'_i$ . (ii) **Consistency:** If every honest party's input is externally valid, then all honest parties output the same value  $v'$ . (iii) **Termination:** If every honest party's input is externally valid, then every honest party  $P_i$  terminates with an output value  $v'_i$ .

**Reliable Broadcast.** A *reliable broadcast (RBC) protocol* [22] allows a designated party  $P_s$  (called sender) to consistently distribute a message among all parties. In contrast to synchronous broadcast, reliable broadcast does not require full termination. We formally define an RBC protocol as follows.

**Definition 3 (RBC Protocol).** Let  $\Pi$  be a protocol executed by  $n$  parties  $P_1, \dots, P_n$ , where a designated sender  $P_s$  holds  $v \in V$  as input. We say that  $\Pi$  is a  $(t_c, n)$ -secure RBC protocol if whenever at most  $t_c$  parties are corrupted the following properties hold. (i) **Validity:** If the sender  $P_s$  is honest and holds  $v$  as input, then every honest party  $P_i$  outputs  $v'_i = v$ . (ii) **Consistency:** All honest parties that output a value output the same value  $v'$ . (iii) **Totality:** If an honest party outputs a value, then every honest party eventually outputs a value.

**Superinvertible Matrices.** A *superinvertible (SI) matrix* of dimension  $(\ell, k)$  with  $k \geq \ell$  [60] is a matrix  $A \in \mathbb{Z}_p^{\ell \times k}$  over some field  $\mathbb{Z}_p$  with the property that each of its  $(\ell \times \ell)$ -dimensional square submatrix  $A_I$  is invertible. Large classes

of superinvertible matrices are given in [18, 57]. Looking ahead, each party  $P_i$  applies the SI matrix  $A$  of dimension  $(\ell, k) := (n - 2t_c, n - t_c)$  to its  $k$  secret shares  $f_1(i), \dots, f_k(i)$  that it received from different completed AVSS sharings. The result is  $\ell$  new secret shares  $r_1(i), \dots, r_\ell(i)$  with the property that if at least  $\ell$  input secrets are independent and uniformly random, then the  $\ell$  output secrets are also guaranteed to be independent and uniformly random.

**Asynchronous Verifiable Secret Sharing.** An *asynchronous verifiable secret sharing (AVSS) scheme* [17, 30] consists of two protocols **Share** and **Rec** which allow a designated dealer to share a secret  $s$  over some field  $\mathbb{Z}_p$  among all parties using Shamir secret sharing. Here, the threshold  $t_r \in [t_c, n - t_c)$  specifies the degree of the shared polynomial  $f$ . In our definition of an AVSS scheme, we require a reconstruction protocol in which parties reconstruct exponentiated evaluations of the polynomial  $f$  at the points  $\{0, 1, \dots, n\}$ . We formally define an AVSS scheme over the group  $(\mathbb{G}, p, g)$  as follows.

**Definition 4** ( $(t_c, t_r, n)$ -Threshold AVSS Scheme). *Let  $\Pi = (\text{Share}, \text{Rec})$  be a pair of protocols executed by  $n$  parties  $P_1, \dots, P_n$ , where a designated dealer  $P_d$  holds a secret  $s \in \mathbb{Z}_p$  as input. Upon completion of **Share** parties only maintain a state and do not output anything. Parties can then call **Rec** with their state and output a tuple of  $n + 1$  elements in  $\mathbb{G}$  and an element in  $\mathbb{Z}_p$ . We say that  $\Pi$  is a complete  $(t_c, t_r, n)$ -threshold AVSS scheme if whenever at most  $t_c$  parties are corrupted the following properties hold:*

- *Correctness. Once the first honest party completes **Share**, there exists a unique polynomial  $f \in \mathbb{Z}_p[X]$  of degree  $t_r$  such that every honest party  $P_i$  upon completing **Rec** outputs an element  $f(i) \in \mathbb{Z}_p$  and the same tuple  $(S, S_1, \dots, S_n)$  of elements in  $\mathbb{G}$  such that  $S = g^{f(0)}$  and  $S_j = g^{f(j)}$  for all  $j \in [n]$ . Further, if  $P_d$  is honest, then it holds that  $f(0) = s$ .*
- *Termination. If  $P_d$  is honest and all honest parties call **Share**, then all honest parties complete **Share**. Further, if all honest parties call **Share** and an honest party completes **Share**, then all honest parties complete **Share**. Finally, if all honest parties call **Rec**, then all honest parties complete **Rec**.*

Hereafter, we write  $\text{AVSS} := (\text{Share}, \text{Rec})$  to denote a generic complete  $(t_c, t_r, n)$ -threshold asynchronous verifiable secret sharing scheme. If AVSS allows for an arbitrary threshold  $t_r \in [t_c, n - t_c)$ , we call it a high-threshold AVSS scheme.

*Remark 1.* In this definition, we leave out a notion of secrecy and postpone it to Sect. 6 instead. We do this for the following reasons. (i) This allows us to provide the reader with a clearer picture of our work and not overload him with several new technical definitions right at the beginning. (ii) Our secrecy definition for an AVSS scheme is strongly motivated by the one we introduce for an ADKG protocol in the next section. As we organize this work according to a top-down structure, it makes more sense to introduce the secrecy notion after that.

**Non-interactive Zero-Knowledge Proofs.** In our AVSS construction, we use non-interactive zero-knowledge (NIZK) proofs [20]. Informally, a non-interactive

proof system for an **NP** relation  $\mathcal{R}$  with respect to a random oracle  $\mathsf{H}$  is a pair of PPT algorithms  $\mathsf{PS} = (\mathsf{PProve}, \mathsf{PVer})$ , where  $\mathsf{PProve}^{\mathsf{H}}$  takes a statement  $x$  and a witness  $w$  with  $(x, w) \in \mathcal{R}$  as input and outputs a proof  $\pi$ , and  $\mathsf{PVer}^{\mathsf{H}}$  takes the statement  $x$  and the proof  $\pi$  as input and decides to accept or reject. Completeness requires that honestly computed proofs for  $(x, w) \in \mathcal{R}$  are accepted, whereas soundness requires that no malicious prover can find an accepting proof for a false statement  $x$ , i.e., a statement such that  $(x, w) \notin \mathcal{R}$  for all  $w$ . Further, zero-knowledge requires that there is a simulator that can simulate proofs without knowing  $w$  by programming the random oracle  $\mathsf{H}$ . Finally, the system is a proof of knowledge, if there is an extractor that can extract the witness from any proof provided by the adversary. To do so, the extractor is allowed to observe the random oracle queries made by the adversary. Our definitions hence model online-extraction, which is reasonable in the algebraic group model. We postpone formal definitions to the full version.

### 3 Packed Asynchronous DKG and Threshold Signatures

In this section, we introduce the notion of a *packed asynchronous distributed key generation (ADKG) protocol* and define our model of syntax and security of a *threshold signature scheme*.

#### 3.1 Packed Asynchronous DKG

In a regular distributed key generation (DKG) protocol, a set of mutually distrusting parties securely establish a public-secret key pair without relying on a trusted dealer. At the end of the protocol, the public key is output in the clear, whereas the secret key is kept as a virtual secret distributed among all parties. This shared secret key can then be used for threshold cryptosystems, such as threshold signatures or threshold encryption, without ever being explicitly reconstructed. When the underlying network is asynchronous, we call it an *asynchronous DKG (ADKG)*. In the following, we introduce and define the notion of an  $(\ell, t_c, t_r, n)$ -*packed ADKG protocol* which allows  $n$  parties out of which at most  $t_c$  are corrupted to generate  $\ell \geq 1$  independent shared keys each with reconstruction threshold  $t_r \in [t_c, n - t_c)$  in a way that is potentially more efficient than just executing  $\ell$  instances of an ADKG protocol in parallel. The basic idea is to realize the same functionality as if  $\ell$  *independent* instances of an ADKG protocol were run in parallel. For the definition, we use the group  $\mathbb{G}$  specified by  $\mathit{par} = (\mathbb{G}, p, g)$ . Hereafter, fix the parameter  $\delta_a := t_r + 1 - t_c$ . The subscript stands for *asynchrony*, since the two thresholds  $t_r$  and  $t_c$  coincide in synchrony.

**Definition 5** ( $(\ell, t_c, t_r, n)$ -**Packed ADKG Protocol**). *Let  $\Pi$  be a protocol executed by  $n$  parties  $P_1, \dots, P_n$ , where for each  $j \in [\ell]$ ,  $P_i$  outputs a secret key share  $r_{j,i}$ , a vector of public key shares  $(R_{j,1}, \dots, R_{j,n})$ , and a public key  $R_j$ . We say that  $\Pi$  is an oracle-aided secure  $(\ell, t_c, t_r, n)$ -packed ADKG protocol if whenever at most  $t_c$  parties are corrupted the following properties hold:*

- Consistency. For each  $j \in [\ell]$ , all honest parties output the same public key  $R_j = g^{x_j}$  and the same vector of public key shares  $(R_{j,1}, \dots, R_{j,n})$ .
- Correctness. For each  $j \in [\ell]$ , there exists a unique polynomial  $f_j \in \mathbb{Z}_p[X]$  of degree  $t_r$  such that for all  $i \in [n]$ ,  $r_{j,i} = f_j(i)$  and  $R_{j,i} = g^{r_{j,i}}$ . Moreover,  $R_j = g^{f_j(0)}$ .
- Termination. If all honest parties participate in the protocol execution, then all honest parties terminate with an output.
- Oracle-aided Simulatability. There exists  $k \in \text{poly}(\lambda)$  with  $k \geq \ell(t_r + 1)$  such that for any PPT algorithm  $A$ , there exists an algebraic PPT simulator  $\text{Sim}$  that on input  $\xi := (g^{z_1}, \dots, g^{z_k}) \in \mathbb{G}^k$  makes  $k' := k - \ell\delta_a$  queries to the oracle  $\text{DL}_{\mathbb{G},g}$  and such that:
  - Syntax.  $\text{Sim}$  simulates the role of the honest parties in an execution of  $\Pi$ . At the end of the simulation,  $\text{Sim}$  outputs the public keys  $R_1, \dots, R_\ell$  and public key shares  $(R_{j,1}, \dots, R_{j,n})$  for all  $j \in [\ell]$ .
  - Queries upon Corruption. Denote by  $\mathcal{C} \subset [n]$  the dynamic set of corrupted parties. Once the first honest party outputs  $(R_{j,1}, \dots, R_{j,n})$  for all  $j \in [\ell]$ , the following holds. Upon corruption query  $i \in [n] \setminus \mathcal{C}$ ,  $\text{Sim}$  invokes  $\text{DL}_{\mathbb{G},g}$  on input  $R_{j,i} = g^{r_{j,i}}$  for all  $j \in [\ell]$  among (possibly) other input elements. Conversely, it does not query  $R_{j,i}$  for any  $j \in [\ell]$  before that corruption.
  - Query Independence. Let  $\mathcal{C}$  be as before and  $\mathcal{H} := [n] \setminus \mathcal{C}$ . Assume that  $|\mathcal{C}| = t_c$  after a simulation of  $\Pi$ . For  $i \in [k - \ell\delta_a]$ , denote by  $g_i \in \mathbb{G}$  the  $i$ -th query to  $\text{DL}_{\mathbb{G},g}$  and let  $(\hat{a}_i, a_{i,1}, \dots, a_{i,k})$  be the corresponding algebraic vector, i.e.,  $g_i = g^{\hat{a}_i} \cdot \xi_1^{a_{i,1}} \cdot \dots \cdot \xi_k^{a_{i,k}}$ . Further, denote by  $(\hat{b}_i, b_{i,1}, \dots, b_{i,k})$  the algebraic vector of the public key share  $R_i$  for all  $\mathbf{i} = (j, i) \in [\ell] \times \mathcal{H}$ . Then for all  $\mathcal{I} := I^\ell \subset \mathcal{H}^\ell$  with  $|I| = \delta_a$ , the following matrix is invertible over  $\mathbb{Z}_p$

$$L(\mathcal{I}, \mathcal{C}) := \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,k} \\ \vdots & \vdots & & \vdots \\ a_{k-\ell\delta_a,1} & a_{k-\ell\delta_a,2} & \cdots & a_{k-\ell\delta_a,k} \\ b_{\mathbf{i}_1,1} & b_{\mathbf{i}_1,2} & \cdots & b_{\mathbf{i}_1,k} \\ \vdots & \vdots & & \vdots \\ b_{\mathbf{i}_{\ell\delta_a},1} & b_{\mathbf{i}_{\ell\delta_a},2} & \cdots & b_{\mathbf{i}_{\ell\delta_a},k} \end{pmatrix} \in \mathbb{Z}_p^{k \times k},$$

where the indices  $\mathbf{i}_{(\cdot)}$  range over the set  $\bigcup_{j \in [\ell]} (\{j\} \times I)$ . Whenever  $\text{Sim}$  completes a simulation of an execution of  $\Pi$ , we call  $L(\mathcal{I}, \mathcal{C})$  the simulatability matrix of  $\text{Sim}$  (for this particular simulation and the set  $\mathcal{I}$ ). Further, we call  $k$  a simulatability factor of  $\Pi$ .

- Bad Event. There is an event  $\text{Bad}$ , such that for any PPT algorithm  $A$ , the probability of  $\text{Bad}$  in an execution of  $\Pi$  with adversary  $A$  is negligible.
- Indistinguishability. Denote by  $\text{view}_{A,\Pi}$  the view of  $A$  in an execution of  $\Pi$ . Denote by  $\text{view}_{A,\xi,\text{Sim}}$  the view of  $A$  when interacting with  $\text{Sim}$  on input  $\xi$ . Then, the distributions  $(\xi, \text{view}_{A,\Pi})$  and  $(\xi, \text{view}_{A,\xi,\text{Sim}})$  where  $\xi \leftarrow_{\mathcal{S}} \mathbb{G}^k$  and both distributions conditioned on  $\neg \text{Bad}$  are statistically close.

For  $\ell = 1$  (when the packing is trivial), we simply call  $\Pi$  a  $(t_c, t_r, n)$ -threshold ADKG protocol (over  $(\mathbb{G}, p, g)$ ). Further, we call  $\ell \geq 1$  the packing parameter.

*Remark 2.* In our above definition of oracle-aided security, we do not require the simulator  $\text{Sim}$  to terminate once it outputs the public keys, but only after it has made the required  $k - \ell\delta_a$  calls to the oracle  $\text{DL}_{\mathbb{G},g}$  (conditioned on the simulation of  $\Pi$  being completed). The reason for this being that adaptive corruptions can happen even after termination of the DKG protocol, e.g., when the DKG is part of a more complex protocol such as a threshold signature scheme.

**Discussion.** For simplicity, we consider only the case  $\ell = 1$  in this discussion. First, note that consistency, correctness, and termination notions are in line with established definitions from the literature for DKG protocols. In addition, our definition is built upon the oracle-aided algebraic security (OAAS) notion from [8] which is defined for DKG protocols with a single threshold  $t$ . We adjust their definition in several ways. First, we extend it to the  $(t_c, t_r)$ -dual-threshold setting which is often relevant in asynchronous networks. Second, we state a more precise requirement on the behavior of the oracle-aided simulator  $\text{Sim}$ , which is explained below. This allows us to make the DKG definition suitable for a more general framework of adaptively secure threshold signatures like threshold Schnorr and threshold BLS.

We begin with our new property “Queries upon Corruption” that specifies  $\text{Sim}$ ’s behavior when a corruption  $i \in \mathcal{H}$  happens after the public key shares are defined from the protocol. Specifically, we require  $\text{Sim}$  to call  $\text{DL}_{\mathbb{G},g}$  on input  $R_i = g^{f(i)}$  only upon that event and not before. The intuition for this being that any reasonable simulator should not know the secret key share  $f(i)$  of that party  $P_i$  before the corruption happens; not surprisingly, all the OAAS simulators constructed in [8] have this property. We proceed with the property “Query Independence” that upon [8] takes a dual-threshold  $(t_c, t_r)$  into consideration. For this, we introduce the set  $\mathcal{I}$ . To understand this, we observe that the idea behind the invertability of the matrix  $L(\mathcal{C})$  as given in their paper is that the joint secret key  $f(0)$  should not be known to the simulator even after  $t_c$  corruptions happened. In the dual-threshold setting, we want this property to hold even if  $t_r - t_c = \delta_a - 1$  additional secret key shares are leaked. That is why we require the algebraic vectors of any  $|I| = \delta_a$  additional public key shares to be independent from already leaked data. Finally, note that [8] does not take computationally indistinguishable simulations into consideration. For better composability, we separate the computational and statistical aspects by introducing the property “Bad Event” and making the property “Indistinguishability” statistical.

### 3.2 Robust Threshold Signatures

In the following, we introduce the syntax and security notions for robust threshold signature schemes. These are in line with established definitions but adopted to the structure of our protocol.



**Syntax and Completeness.** In our model, a threshold signature scheme has the following structure. First, all parties  $P_1, \dots, P_n$  run a regular  $(t_c, t_r, n)$ -threshold ADKG protocol denoted by IDKG (called the *initial ADKG*). Having done this, each party  $P_i$  holds a secret key share  $\text{sk}_i$  and the public key shares  $\text{pk}_1, \dots, \text{pk}_n$  of other parties along with the public key  $\text{pk}$ . Following this, parties repeatedly run two parallel instances of an  $(\ell, t_c, t_r, n)$ -packed ADKG protocol denoted by NDKG in the background. The keys generated by these executions are interpreted as nonces. In particular, after each parallel execution of the *Nonce-ADKG* protocol NDKG, the parties obtain  $2\ell$  new and independent nonces. To simplify matters, we assume that the nonces are output in pairs  $(R_j, R'_j)$ . For each such public nonce  $R_j$  (respective  $R'_j$ ), each party  $P_i$  also obtains its secret nonce share  $r_{j,i}$  (respective  $r'_{j,i}$ ) along with the public nonce shares  $(R_{j,1}, \dots, R_{j,n})$  (respective  $(R'_{j,1}, \dots, R'_{j,n})$ ) of other parties. For signing, we adapt the double-nonce approach introduced by Komlo and Goldberg [65] in order to prevent concurrent session attacks [19, 47]<sup>5</sup>. That is, we assume that parties have agreement on a previously generated but never before used nonce pair  $(R_j, R'_j)$  and use the effective nonce  $\hat{R}_j = R_j R'_j{}^b$  to sign a message  $m$  where the scalar  $b \in \mathbb{Z}_p$  is derived from a random oracle  $\text{H}_{\text{non}}$  as  $b = \text{H}_{\text{non}}(\text{pk}, R_j, R'_j, m)$ . Upon such a signing request, each party derives the effective nonce shares  $(\hat{R}_{j,1}, \dots, \hat{R}_{j,n})$  and its effective secret nonce share  $\hat{r}_{j,i}$  analogously. In this light, the protocol essentially becomes *non-interactive*: When party  $P_i$  wants to sign message  $m$  with respect to effective nonce  $\hat{R}_j$ , it runs an algorithm  $\text{SSign}$  using its secret key  $\text{sk}_i$  and its secret nonce share  $\hat{r}_{j,i}$  on message  $m$ . As a result, the party obtains a signature share  $\sigma_i$  that it sends to all other parties. This signature share can be verified with respect to the parties public key share  $\text{pk}_i$  and the public nonce share  $\hat{R}_{j,i}$ . Upon receiving  $t_r + 1$  valid signature shares, a party can locally combine them into a full signature  $\sigma$  on  $m$  with randomness  $\hat{R}_j$ . This signature can now be verified with respect to the public key  $\text{pk}$  only. From this explanation of the execution model, it is clear that we can define such a threshold signature scheme by specifying the initial ADKG protocol, the packed ADKG protocol for nonce generation, and algorithms for signing and verification similar to a non-interactive threshold signature scheme [8, 67].

**Definition 6 (Threshold Signature Scheme).** An  $(\ell, t_c, t_r, n)$ -threshold signature scheme is a tuple of PPT protocols and algorithms  $\Sigma = (\text{IDKG}, \text{NDKG}, \text{SSign}, \text{SVer}, \text{Comb}, \text{Ver})$  with the following syntax:

- IDKG: This is a  $(t_c, t_r, n)$ -threshold asynchronous DKG protocol as specified in Definition 5.
- NDKG: This is an  $(\ell, t_c, t_r, n)$ -packed asynchronous DKG protocol as specified in Definition 5.
- SSIGN: The signature share generation algorithm takes as input a secret key share  $\text{sk}_i \in \mathbb{Z}_p$ , a public key  $\text{pk} \in \mathbb{G}$ , two secret nonce shares  $r_i, r'_i \in \mathbb{Z}_p$ , two

<sup>5</sup> Nick et al. [70] introduced essentially the same technique at the same time to construct a two-round Schnorr multi-signature with a rigorous security analysis.

- public nonces  $R, R' \in \mathbb{G}$ , and a message  $m \in \{0, 1\}^*$ . It outputs a signature share  $\sigma_i$ .
- **SVer**: The signature share verification algorithm takes as input a public key  $\mathbf{pk} \in \mathbb{G}$ , a public key share  $\mathbf{pk}_i \in \mathbb{G}$ , two public nonces  $R, R' \in \mathbb{G}$ , two public nonce shares  $R_i, R'_i \in \mathbb{G}$ , a message  $m \in \{0, 1\}^*$ , and a signature share  $\sigma_i$ . It outputs 1 (accept) or 0 (reject).
  - **Comb**: The signature share combining algorithm takes as input two public nonces  $R, R' \in \mathbb{G}$ , a message  $m \in \{0, 1\}^*$ , and a set  $\mathcal{S}$  of  $t_r + 1$  signature shares  $(\sigma_i, i)$  with corresponding indices. It outputs either a signature  $\sigma$  or  $\perp$ .
  - **Ver**: The signature verification algorithm takes as input a public key  $\mathbf{pk} \in \mathbb{G}$ , a message  $m \in \{0, 1\}^*$ , and a signature  $\sigma$ . It outputs 1 (accept) or 0 (reject).

Further, we require  $\Sigma$  to satisfy straightforward correctness properties which are specified in the full version (Correctness of TSS).

We emphasize that our definition models a *robust* threshold signing protocol [55]. The reason for this is that the protocol NDKG terminates with distributed nonces each having reconstruction threshold  $t_r$ . Since there are at least  $n - t_c \geq t_r + 1$  honest parties in the system, it is guaranteed for every honest party to obtain enough valid signature shares (even if no corrupt party sends a valid signature share or anything at all) and thus to compute the full signature.

**Security Model.** We define the security of a threshold signature scheme following our syntax. The established security definition for non-interactive adaptively secure threshold signatures [8, 67] allows the adversary to adaptively ask for signature shares and corruptions for up to  $t_c$  parties of its choice. In the end, the adversary succeeds if it outputs a message  $m^*$  and a valid signature  $\sigma^*$  for it such that it obtained at most  $t_c$  signature shares for  $m^*$ . In the synchronous setting, the thresholds for corruption and reconstruction coincide. As we work in an asynchronous network, we adjust their definition to a dual-threshold: the protocol should be resistant against  $t_c$  corruptions while providing security for even up to  $t_r$  leaked signature shares. Additionally, we let the adversary freely decide when parties execute a new (parallel) instance of the Nonce-ADKG protocol in which he also participates. Finally, signature shares are generated with respect to a specific nonce pair that has been generated but not used previously and is specified by the adversary.

**Definition 7 (Unforgeability Under Chosen Message Attack).** Let  $\Sigma = (\text{IDKG}, \text{NDKG}, \text{SSign}, \text{SVer}, \text{Comb}, \text{Ver})$  be an  $(\ell, t_c, t_r, n)$ -threshold signature scheme. For an algorithm  $A$ , we consider the following experiment:

1. **Setup.** Initialize a corruption set  $\mathcal{C} := \emptyset$  and a signing query set  $\mathcal{Q} := \emptyset$ . For each party  $P_i$ ,  $i \in [n]$ , initialize an empty state  $St_i$ . Run  $A$  on input  $\text{par}$ . At any point throughout the experiment,  $A$  can issue corruption queries by submitting an index  $i \in [n] \setminus \mathcal{C}$ . In this case, update  $\mathcal{C} := \mathcal{C} \cup \{i\}$  and return the internal state  $St_i$  of party  $P_i$  to  $A$ . Henceforth,  $A$  fully controls  $P_i$ .

2. Initial Asynchronous DKG. *Initiate an execution of IDKG among parties  $P_1, \dots, P_n$ , where at any point in time,  $A$  controls all parties  $P_i$  with  $i \in \mathcal{C}$ , and the experiment simulates all other parties following the protocol, and adds their respective state to  $St_i$ . Denote by  $\text{pk}$  and  $(\text{pk}_1, \dots, \text{pk}_n)$  the public key and public key shares determined by IDKG. Denote by  $\text{sk}_i$  for all  $i \in [n] \setminus \mathcal{C}$  the secret key shares of the honest parties. When the execution of IDKG has terminated, add  $\text{sk}_i$  to  $St_i$  for all  $i \in [n] \setminus \mathcal{C}$ .*
3. Online Phase. *During this phase,  $A$  gets additional access to oracles that answer queries of the following types:*
  - Nonce-ADKG Query. *When  $A$  queries this oracle, a new parallel protocol execution<sup>6</sup> of NDKG among the parties  $P_1, \dots, P_n$  is initiated. As for the initial distributed key generation,  $A$  controls all parties  $P_i$  with  $i \in \mathcal{C}$ , and the experiment simulates all other parties following the protocol, and adds their respective state to  $St_i$ . When this protocol terminates for the  $(k+1)$ -th time, let  $(R_{k\ell+1}, R'_{k\ell+1}), \dots, (R_{k\ell+\ell}, R'_{k\ell+\ell})$  be the respective public nonce pairs, and let  $R_{k\ell+j,1}, \dots, R_{k\ell+j,n}$  and  $R'_{k\ell+j,1}, \dots, R'_{k\ell+j,n}$  for each  $j \in [\ell]$  be the respective public nonce shares. Further, for each party  $P_i$  with  $i \in [n] \setminus \mathcal{C}$ , let  $(r_{k\ell+1,i}, r'_{k\ell+1,i}), \dots, (r_{k\ell+\ell,i}, r'_{k\ell+\ell,i})$  be the respective secret nonce share pairs that party  $P_i$  obtains. These secret nonce share pairs are added to  $St_i$ .*
  - Signing Query. *When  $A$  submits a new tuple  $(i, j, m) \notin \mathcal{Q}$  for an  $i \in [n] \setminus \mathcal{C}$  and nonce index  $j$  such that  $(R_j, R'_j)$  is defined, then: If there is an  $m' \neq m$  and an  $i' \in [n]$  such that  $(i', j, m') \in \mathcal{Q}$ , then return  $\perp$ . Otherwise, set  $\mathcal{Q} := \mathcal{Q} \cup \{(i, j, m)\}$  and return  $\sigma \leftarrow \text{SSign}(\text{sk}_i, \text{pk}, r_{j,i}, r'_{j,i}, R_j, R'_j, m)$ .*
4. Winning Condition. *When  $A$  outputs a message  $m^*$  and a signature  $\sigma^*$ , let  $\mathcal{S} \subset [n]$  denote the subset of parties for which  $A$  made a signing query for  $m^*$ , i.e., let*

$$\mathcal{S} := \{i \in [n] \mid \exists j \text{ s.t. } (i, j, m^*) \in \mathcal{Q}\}.$$

*Return 1 if  $|\mathcal{C}| \leq t_c$ ,  $|\mathcal{C} \cup \mathcal{S}| \leq t_r$ , and  $\text{Ver}(\text{pk}, m^*, \sigma^*) = 1$ . Else, return 0.*

*We say that  $\Sigma$  is unforgeable under chosen message attacks (or UF-CMA secure) if for any PPT algorithm  $A$ , the probability that the above experiment outputs 1 is negligible in  $\lambda$ .*

*Remark 3.* In our security model, we assume that parties agree on which nonce to use for which message (similar to a session identifier). The reason for this is to make our reduction in the security proof for our protocol (cf. Theorem 1) to go through. We note that it is also not clear to us if the protocol remains secure otherwise. Interestingly, the reductions in prior works [18, 37, 38, 73] also seem to rely on this assumption without explicitly stating it in their security model; however, the works [57, 77] explicitly state this requirement in their security model and mention that on a distributed system signing requests must go through a consensus mechanism anyway.

<sup>6</sup> By a parallel execution, we refer to a pair of instances of NDKG.

## 4 Robust Threshold Schnorr Signatures

In this section, we provide a generic construction for a high-threshold, robust, and efficient threshold Schnorr signature scheme and analyze its security.

### 4.1 Our Construction

In the following, we give a generic construction for a robust threshold Schnorr signature scheme (also refer to Fig. 2). Our construction is based on the technique introduced by Gennaro et al. [54, 55]. In their work, they observed that in order to obtain a robust threshold Schnorr signature, the nonce itself should be computed in a distributed threshold fashion realized via a DKG protocol. Building upon this idea, we implement the DKG protocol for nonce generation with a packed ADKG protocol. For this, let  $\ell, t_c, t_r, n \in \mathbb{N}$  be natural numbers such that  $t_c < n/3$  and  $t_r \in [t_c, n - t_c]$ .

**Construction.** Let IDKG be a  $(t_c, t_r, n)$ -threshold ADKG protocol and let NDKG be an  $(\ell, t_c, t_r, n)$ -packed ADKG protocol. Further, let  $H, H_{\text{non}}: \{0, 1\}^* \rightarrow \mathbb{Z}_p$  be two hash functions (modeled as random oracles). Then, the threshold Schnorr signature scheme  $\text{SchnorrTS}[\text{IDKG}, \text{NDKG}] = (\text{IDKG}, \text{NDKG}, \text{SSign}, \text{SVer}, \text{Comb}, \text{Ver})$  is defined as follows:

- $\text{SSign}(\text{sk}_i, \text{pk}, r_i, r'_i, R, R', m)$ : Compute  $b := H_{\text{non}}(\text{pk}, R, R', m)$  and the effective nonce  $\hat{R} := RR'^b$ . Further, compute  $\hat{r}_i := r_i + b \cdot r'_i$  and  $c := H(\text{pk}, \hat{R}, m)$ . Return the signature share  $\sigma_i := c \cdot \text{sk}_i + \hat{r}_i \in \mathbb{Z}_p$ .
- $\text{SVer}(\text{pk}, \text{pk}_i, R, R', R_i, R'_i, m, \sigma_i)$ : Compute  $b := H_{\text{non}}(\text{pk}, R, R', m)$ , the effective nonce  $\hat{R} := RR'^b$ , the effective nonce share  $\hat{R}_i := R_i R'_i{}^b$ , and further  $c := H(\text{pk}, \hat{R}, m)$ . Return 1 if  $\text{pk}_i^c \cdot \hat{R}_i = g^{\sigma_i}$  and 0 otherwise.
- $\text{Comb}(R, R', m, \mathcal{S})$ : Parse  $\mathcal{S}$  as a set of  $t_r + 1$  signature shares  $(\sigma_i, i)$  with corresponding indices. Denote the set of these indices by  $\mathcal{I}$ . Compute  $s := \sum_{i \in \mathcal{I}} L_{i, \mathcal{I}} \sigma_i$  where  $L_{i, \mathcal{I}}$  denotes the  $i$ -th Lagrange coefficient for the set  $\mathcal{I}$ . Further, compute the effective nonce  $\hat{R} := RR'^b$  where  $b := H_{\text{non}}(\text{pk}, R, R', m)$ . Return the signature  $\sigma := (\hat{R}, s)$ .
- $\text{Ver}(\text{pk}, m, \sigma)$ : Parse  $\sigma$  as  $\sigma = (\hat{R}, s)$ . Return 1 if  $\text{pk}^c \cdot \hat{R} = g^s$  and 0 otherwise.

### 4.2 Security Analysis

We proceed with the security proof of  $\text{SchnorrTS}[\text{IDKG}, \text{NDKG}]$  assuming oracle-aided security of IDKG and NDKG. For this, we give a security reduction from the hardness of the OMDL assumption to the unforgeability (cf. Definition 7) of our threshold signature scheme  $\text{SchnorrTS}[\text{IDKG}, \text{NDKG}]$ .

**Proof Intuition.** We give here an intuition for our proof. The key idea of our reduction is to embed the OMDL challenge  $\xi$  into the public keys  $\text{pk}_1, \dots, \text{pk}_n$  of parties that are output by IDKG and into the public nonces  $\{R_i, R'_i \mid i \in [\ell q_r]\}$  that are output by the  $q_r$  parallel executions of NDKG. Recall that each parallel execution outputs  $2\ell$  nonces that we interpret as  $\ell$  nonce pairs. In order to do so,

we employ the oracle-aided simulators  $\text{Sim}_0$  for IDKG and  $\text{Sim}_j, \text{Sim}'_j$  for the  $j$ -th parallel execution  $\text{NDKG}_j$  of NDKG. Corruption queries  $i \in \mathcal{H}$  are handled by  $\text{Sim}_0$  to return its secret key share  $\text{sk}_i$  (along with other internal data generated from IDKG related to  $P_i$ ), and by  $\text{Sim}_j, \text{Sim}'_j$  to return its  $2\ell$  secret nonce shares from  $\text{NDKG}_j$  (along with other internal data generated from  $\text{NDKG}_j$  related to  $P_i$ ). Signature share queries  $(i, j, m)$  for an honest party  $P_i$  and (previously generated) nonce pair  $(R_j, R'_j)$  are handled in one of two ways. (i) If the reduction already knows  $t_r + 1$  signature shares for  $(j, m)$ <sup>7</sup>, then it computes the remaining shares by Lagrange interpolation and returns the signature share  $\sigma_{j,i}$  of that party. (ii) If the reduction knows  $t_r$  or less signature shares for  $(j, m)$ , then it calls the discrete logarithm oracle  $\text{DL}_{\mathbb{G},g}$  on input  $\text{pk}^{c_j} \cdot \hat{R}_{j,i}$  to obtain  $\sigma_{j,i} := c_j \cdot \text{sk}_i + \hat{r}_{j,i}$  and returns it. Here, it can derive the values  $c_j := \text{H}(\text{pk}, \hat{R}_j, m)$ ,  $\hat{R}_{j,i} := R_{j,i} R_{j,i}^{b_j}$ , and  $b_j := \text{H}_{\text{non}}(\text{pk}, R_j, R'_j, m)$  by itself from local computations and consistent lazy sampling for random oracle outputs (if not yet defined).

However, this approach has the subtlety that it exceeds the number of allowed calls to  $\text{DL}_{\mathbb{G},g}$ . If the adversary  $A$  makes a signature share query  $(i, j, m)$  and later in the course of the protocol execution corrupts that same party  $P_i$ , then the reduction would have used  $\text{DL}_{\mathbb{G},g}$  too often: once for  $\text{Sim}_0$  to return the secret key share  $\text{sk}_i$ , once for  $\text{Sim}_j$  to return the secret nonce share  $r_{j,i}$ , once for  $\text{Sim}'_j$  to return the secret nonce share  $r'_{j,i}$ , and once to compute the signature share  $\sigma_{j,i}$ . On the other hand, the identity  $\sigma_{j,i} = c_j \cdot \text{sk}_i + \hat{r}_{j,i}$  tells us that three calls are enough to derive those four values. To make use of this, we carefully leverage the “queries upon corruption” property of the simulators  $\text{Sim}_j, \text{Sim}'_j$  for  $j \geq 1$ . More precisely, as we know that  $\text{Sim}_j$  queries the discrete logarithm oracle on the element  $R_{j,i}$  upon corruption of party  $P_i$ , we simply answer this query on  $R_{j,i}$  by computing  $\hat{r}_{j,i} := \sigma_{j,i} - c_j \cdot \text{sk}_i$  and returning the value  $r_{j,i} = \hat{r}_{j,i} - b_j r'_{j,i}$  directly instead of calling  $\text{DL}_{\mathbb{G},g}$ . In particular, we avoid redundant calls to  $\text{DL}_{\mathbb{G},g}$ . At the end of the game, we obtain a forgery  $(m^*, \sigma^*)$  from  $A$  which we convert into a solution of the OMDL challenge  $\xi$ ; recall that  $\sigma^*$  is of the form  $(R^*, s^*)$ . This is done as follows. First, as  $A$  is an algebraic adversary, it returns the random oracle query  $\text{H}(\text{pk}, R^*, m^*)$  together with a representation of elements in  $\mathbb{Z}_p$ . Second, using the forgery  $(m^*, \sigma^*)$ , known signature shares  $\{\sigma_{j,1}, \dots, \sigma_{j,n}\}_j$ , and known secret key shares  $\text{sk}_i$  from  $t_r$  parties, we can compute the secret key  $\text{sk}$ . Third, this allows us to compute all secret key shares  $\text{sk}_1, \dots, \text{sk}_n$  and thus using the signature shares also all secret nonce shares  $\{\hat{r}_{j,1}, \dots, \hat{r}_{j,n}\}_j$ . Finally, by inverting the simulatability matrices of all oracle-aided simulators  $\text{Sim}_0, \text{Sim}_1, \dots$ , we can translate the aforementioned values into an OMDL solution. We provide a full proof of the following theorem in the full version.

**Theorem 1 (ADKG  $\longrightarrow$  Threshold Schnorr).** *Let  $\ell, t_c, t_r, n \in \mathbb{N}$  be natural numbers such that  $t_c < n/3$  and  $t_r \in [t_c, n - t_c]$ . Let IDKG be an algebraic<sup>8</sup> oracle-aided secure  $(t_c, t_r, n)$ -threshold ADKG protocol and let NDKG be*

<sup>7</sup> Recall that in our model, a message  $m \in \{0, 1\}^*$  is always signed with respect to a previously generated and agreed-upon nonce pair  $(R_j, R'_j)$ . That is, when message  $m$  is signed, the parties have agreement on which nonce index  $j$  to use for it.

<sup>8</sup> That is, all parties behave algebraically and can be modeled as algebraic machines.

an algebraic oracle-aided secure  $(\ell, t_c, t_r, n)$ -packed ADKG protocol. Further, let  $H, H_{non}: \{0, 1\}^* \rightarrow \mathbb{Z}_p$  be two random oracles. Then, the threshold Schnorr signature scheme  $\text{SchnorrTS}[\text{IDKG}, \text{NDKG}]$  (cf. Sect. 4.1) is UF-CMA secure in the algebraic group model under the OMDL assumption.

## 5 Efficient Packed ADKG Protocol

In this section, we provide a generic construction for a packed ADKG protocol that is more efficient than naively executing many instances of a regular ADKG protocol in parallel.

### 5.1 Our Construction

In the following, we give a construction PADKG of an  $(\ell, t_c, t_r, n)$ -packed ADKG protocol over  $(\mathbb{G}, p, g)$  where  $\ell = n - 2t_c$  and  $t_r \in [t_c, n - t_c]$  is arbitrary. Our protocol PADKG relies on the following building blocks: (i) a high-threshold AVSS scheme AVSS, (ii) an MVBA protocol MVBA with external validity function `checkValidity` (cf. Eq. 1 below), and (iii) a bulletin board PKI (cf. Sect. 2). We give an informal description of the protocol and refer to the full version for a formal description as pseudocode.

**Packed ADKG Description.** Conceptually, parties agree on  $n - t_c$  AVSS sharings and use a superinvertible (SI) matrix to extract as much randomness from these as possible. In more detail, our protocol has the following four steps:

1. **Sharing.** In the first step, each party  $P_i$  shares a secret  $s_i \leftarrow_{\$} \mathbb{Z}_p$  via AVSS. This means that  $s_i$  lies on a polynomial  $f_i \in \mathbb{Z}_p[X]$  of degree  $t_r$ . Afterwards, each party waits for  $n - t_c$  AVSS sharings to complete locally and stores the corresponding indices of the dealers in a set  $\text{dealers}_i$ . Since the network is asynchronous, each party might have a different set  $\text{dealers}_i$  of locally completed sharings. Therefore, parties need to agree on exactly one such set using an MVBA protocol MVBA. However, the problem is that these sets as are cannot be checked via an *external validity function* which is needed for the MVBA protocol. This issue is resolved as follows.
2. **MVBA Execution.** Once its set  $\text{dealers}_i$  of completed sharings reaches size  $n - t_c$ , party  $P_i$  sends it as a proposal  $\text{prop}_i$  to all other parties with the aim to collect at least  $t_c + 1$  signatures from other parties on it that it stores in a set  $\text{sigs}_i$ . Conversely, a party  $P_j$  only issues a signature on  $\text{prop}_i$  once all AVSS sharings specified by  $\text{prop}_i$  have completed at  $P_j$  itself. Once the set  $\text{sigs}_i$  of collected signatures on  $\text{prop}_i$  reaches size  $t_c + 1$  (guaranteeing that these sharings completed at an honest party and thus by completeness of AVSS eventually also at all other honest parties), party  $P_i$  invokes MVBA on input  $(\text{prop}_i, \text{sigs}_i)$  with external validity function `checkValidity` given by:

$$(|\text{prop}| = n - t_c) \wedge (|\text{sigs}| \geq t_c + 1) \wedge (\forall (j, \sigma_j) \in \text{sigs} : \text{Ver}(\text{vk}_j, \text{prop}, \sigma_j) = 1). \quad (1)$$

3. **AVSS Reconstruction.** Once the MVBA terminates and parties have agreement on a set `dealers` of  $n - t_c$  dealers whose sharings completed (we assume that parties order the set `dealers` by increasing party index), parties proceed with the (possibly interactive) reconstruction phase. The result is that each party  $P_i$  obtains a vector  $(S_j, S_{j,1}, \dots, S_{j,n})$  of group elements in  $\mathbb{G}$  such that  $S_{j,k} = g^{f_j(k)}$  for  $k \in [n]$  along with its secret share  $s_{j,i} = f_j(i)$  for all  $j \in \text{dealers}$ . While this phase comes for free for some AVSS schemes [44], it is required for other schemes, e.g., those that build upon KZG commitments [2]. This phase comes for free with our AVSS scheme in Sect. 6.1.
4. **SI Matrix Application.** Having done this, each party  $P_i$  locally applies (i.e., matrix multiplication from the left) the  $(\ell, n - t_c)$ -dimensional superinvertible matrix `SI` to its secret shares arranged in a vector  $(s_{j,i})_{j \in \text{dealers}}$  to obtain an  $\ell$ -dimensional vector  $(r_{1,i}, \dots, r_{\ell,i})$  of new private outputs. Additionally,  $P_i$  applies `SI` in the exponent to the matrix with rows  $(S_j, S_{j,1}, \dots, S_{j,n})$  for  $j \in \text{dealers}$  to obtain an  $n$ -dimensional vector  $(R_j, R_{j,1}, \dots, R_{j,n})$  of new public outputs for each  $j \in [\ell]$ . Looking ahead, these vectors constitute the public nonces and public nonce shares. These operations are captured by the algorithm `ApplySI`. At the end of this phase, each party  $P_i$  outputs a set  $\{(j, r_{j,i}, (R_j, R_{j,1}, \dots, R_{j,n}))\}_{j \in [\ell]}$ . For each  $j \in [\ell]$ ,  $R_j$  is the  $j$ -th public nonce with corresponding public shares  $(R_{j,1}, \dots, R_{j,n})$  of all parties and  $r_{j,i}$  is party  $P_i$ 's secret share of the nonce  $R_j$ .

The idea of the final phase is the following. Only  $\ell = n - 2t_c$  of the polynomials  $f_j$  shared by the parties  $j \in \text{dealers}$  are guaranteed to be chosen from honest parties and thus uniformly random. By taking  $\ell$  linearly independent linear combinations specified by the superinvertible matrix `SI`, parties obtain  $\ell$  new polynomials  $r_1, \dots, r_\ell \in \mathbb{Z}_p[X]$  of degree  $t_r$  shared among them that are guaranteed to be uniformly random and hidden from the adversary. By applying the `SI` matrix also to public output related to  $f_j$  (i.e., the public elements  $S_j, S_{j,1}, \dots, S_{j,n}$ ) for all  $j \in \text{dealers}$ , parties obtain regular Feldman commitments to the polynomials  $r_1, \dots, r_\ell$  (i.e., the public elements  $R_j, R_{j,1}, \dots, R_{j,n}$  for each polynomial  $r_j$ ,  $j \in [\ell]$ ) which makes subsequent threshold signing for Schnorr signatures in the high-threshold setting possible and efficient.

## 5.2 Security Analysis

We proceed with the security analysis of our generic packed ADKG protocol `PADKG` described before. For this, we introduce a security notion for AVSS schemes that is very similar to the oracle-aided simulatability notion for packed ADKG (cf. Definition 5). Due to the similarity, we provide the definition in the full version. Then, we show that this notion for the AVSS in combination with the (regular) security of the MVBA are sufficient to obtain an oracle-aided secure packed ADKG protocol as given in the full version. Here, we emphasize that the proof crucially relies on the defining property of a superinvertible matrix which is that any square submatrix of appropriate size is invertible.

**Proof Intuition.** In the following, we describe how to construct an oracle-aided simulator  $\text{Sim}$  for PADKG given oracle-aided simulators for AVSS. For this informal overview, we omit the discussion on correctness, consistency, and termination for PADKG, since these follow from standard considerations. For each  $i \in [n]$ , denote by  $\text{Sim}_i$  the simulator for the instance  $\text{AVSS}_i$  with dealer  $P_i$ . Assume that AVSS has simulatability factor  $k$ . Then, our simulator  $\text{Sim}$  has simulatability factor  $kn$ . Let  $\xi := (\xi_1, \dots, \xi_n)$  be elements where  $\xi_i := (\xi_{i,1}, \dots, \xi_{i,k}) \in \mathbb{G}^k$  such that  $\xi_{i,j} = g^{z_{i,j}}$  for some  $z_{i,j} \in \mathbb{Z}_p$ . On input  $(\text{par}, \xi)$ , our simulator  $\text{Sim}$  does the following in an execution of PADKG described by its four phases (cf. Sect. 5.1). First, in the *AVSS sharing* phase, it runs  $\text{Sim}_i$  on input  $(\text{par}, \xi_i)$  for all  $i \in [n]$ . Second, in the *MVBA execution* phase, it runs the protocol faithfully on behalf of all honest parties. Third, the *AVSS reconstruction* phase is also handled by the simulators  $\text{Sim}_i$  for  $i \in [n]$ . Finally, in the *SI matrix application* phase,  $\text{Sim}$  applies SI to the reconstructed vectors from the instances  $\text{AVSS}_i$  for all  $i \in \text{dealers}$  to obtain  $(R_{j,1}, \dots, R_{j,n})$  for  $j \in [\ell]$ . As a result,  $\text{Sim}$  can output all necessary group elements in the group  $\mathbb{G}$  and terminate the protocol.

Throughout the simulation up until the point in which the first honest party outputs the elements  $\{(R_{j,1}, \dots, R_{j,n})\}_{j \in [\ell]}$ , a corruption query  $l \in \mathcal{H}$  is forwarded to all  $\text{Sim}_i$ ,  $i \in [n]$ , simultaneously. In that case, to answer discrete logarithm oracle queries from  $\text{Sim}_i$  on an element  $h' \in \mathbb{G}$ ,  $\text{Sim}$  forwards it to its oracle  $\text{DL}_{\mathbb{G},g}$  and returns the result to  $\text{Sim}_i$ . However, once the event happens in which the first honest party outputs  $\{(R_{j,1}, \dots, R_{j,n})\}_{j \in [\ell]}$ , subsequent corruption queries have to be answered differently<sup>9</sup>. The subtle reason for this is that  $\text{Sim}$  otherwise would make redundant calls to its oracle  $\text{DL}_{\mathbb{G},g}$ , thus violating the required notion of “query independence”. We see this as follows. Assuming party  $P_l$  gets corrupted, the notion of oracle-aided simulatability for (packed) ADKG requires the simulator  $\text{Sim}$  to query  $\text{DL}_{\mathbb{G},g}$  on input  $R_{j,l}$  for all  $j \in [\ell]$  at this point in time. On the other hand, by the the notion of oracle-aided simulatability for AVSS we also know that all simulators  $\text{Sim}_i$  for  $i \in [n]$  will query the discrete logarithm oracle on input  $S_{i,l}$  for all  $i \in [n]$  at this point in time. By definition of  $R_{j,l}$ , we know that for all  $j \in [\ell]$ ,

$$r_{j,l} = m_{j,1}s_{1,l} + \dots + m_{j,n-t_c}s_{n-t_c,l}, \quad (\heartsuit)$$

where the  $m_{j,i} \in \mathbb{Z}_p$  are the entries of the superinvertible matrix  $\text{SI} := (m_{j,i})_{j,i}$  and we assume for simplicity that  $\text{dealers} = \{1, \dots, n - t_c\}$ . Obviously, the required calls from  $\text{Sim}$  to its oracle  $\text{DL}_{\mathbb{G},g}$  on input  $R_{j,l}$ ,  $j \in [\ell]$ , cannot be independent from the ones the individual simulators  $\text{Sim}_i$  would make on input  $S_{i,l}$ ,  $i \in [n]$ . In order to deal with this issue, the simulator  $\text{Sim}$  has to return the values  $s_{i,l} = \text{DL}_{\mathbb{G},g}(S_{i,l})$  to  $\text{Sim}_i$  for all  $i \in [l]$  differently. Concretely, it will first query  $\text{DL}_{\mathbb{G},g}(R_{j,l})$  for all  $j \in [\ell]$  to obtain the values  $\{r_{j,l} \mid j \in [\ell]\}$ . Next, it will choose a random subset  $\mathcal{S} \subset \text{dealers} \setminus \mathcal{C}$  of size  $t_c - |\mathcal{C} \cap \text{dealers}|$  and query  $\text{DL}_{\mathbb{G},g}(S_{i,l})$  for all  $i \in \mathcal{S}$  to obtain a total of  $t_c$  values  $\{s_{i,l} \mid i \in \mathcal{S} \cup (\mathcal{C} \cap \text{dealers})\}$ . From knowledge of these values, the identities in  $(\heartsuit)$ , and the property of SI,

<sup>9</sup> Note that this can only happen under adaptive corruptions.



Sim can compute the remaining values  $s_{i,l}$  from ( $\heartsuit$ ) by inverting a suitable sub-matrix of SI and return these values to the simulators  $\text{Sim}_i$ . Still, special care has to be taken as the set  $\mathcal{C}$  of corrupt parties is dynamically increasing and we have to make the counting argument of calls to  $\text{DL}_{\mathbb{G},g}$  rigorous. We provide a full proof of the following theorem in the full version.

**Theorem 2 (AVSS  $\rightarrow$  ADKG).** *Let  $t_c, t_r, n \in \mathbb{N}$  be natural numbers such that  $t_c < n/3$  and  $t_r \in [t_c, n - t_c)$ . Let AVSS be an oracle-aided secure  $(t_c, t_r, n)$ -threshold AVSS scheme and let MVBA be a  $(t_c, n)$ -secure MVBA protocol. Further, let SI be a superinvertible matrix over  $\mathbb{Z}_p$  of dimension  $(n - 2t_c, n - t_c)$ . Then, PADKG is an oracle-aided secure  $(\ell, t_c, t_r, n)$ -packed ADKG protocol with  $\ell = n - 2t_c$ .*

*Remark 4.* We note that our proof does not rely on the algebraic group model. However, if the AVSS scheme is algebraic (i.e., all parties behave algebraically) and the adversary is algebraic, then all our reductions are also algebraic.

## 6 High-Threshold AVSS Scheme

In this section, we design a new high-threshold AVSS scheme and show that it satisfies our notion of oracle-aided simulatability for AVSS under adaptive corruptions.

### 6.1 Our Construction

We construct a simple high-threshold AVSS scheme  $\text{HAVSS} = (\text{HAVSS.Share}, \text{HAVSS.Rec})$ , relying on bivariate polynomials and NIZK proofs for inner product relations. We provide here an informal description and refer to the full version for formal descriptions as pseudocode.

**Building Blocks.** For the construction, we assume additional  $t_c + 1$  random generators  $g_0, g_1, \dots, g_{t_c} \leftarrow^{\$} \mathbb{G}$ . These can for example be derived from a random oracle. We also assume a non-interactive proof system (see full version for a definition)  $\text{PS}_{\text{open}} = (\text{PProve}_{\text{open}}, \text{PVer}_{\text{open}})$  for the relation

$$\mathcal{R}_{\text{open}} := \left\{ \left( (g, g_0, \dots, g_{t_c}, \text{cm}_i, \omega, y), C_i \right) \mid \text{cm}_i = \prod_{j=0}^{t_c} g_j^{c_j, i} \wedge C_i(\omega) = y \right\},$$

and a non-interactive proof system  $\text{PS}_{\text{exp}} = (\text{PProve}_{\text{exp}}, \text{PVer}_{\text{exp}})$  for the relation

$$\mathcal{R}_{\text{exp}} := \left\{ \left( (g, g_0, \dots, g_{t_c}, \text{cm}_i, \omega, Y), C_i \right) \mid \text{cm}_i = \prod_{j=0}^{t_c} g_j^{c_j, i} \wedge Y = g^{C_i(\omega)} \right\}.$$

Note that both relations are inner-product relations, as evaluating a polynomial at a known location is an inner product. For simplicity, we write them using the

same random oracle  $H$ , while formally we should understand this as two separate random oracles. Further, we omit the elements  $g, g_0, \dots, g_{t_c}$  from the statements to avoid clutter. They are always clear from the context.

Finally, we make use of two deterministic algorithms `Interpolate` and `ExplInterpolate`, which is Lagrange interpolation and Lagrange interpolation in the exponent, respectively. In more detail, these algorithms work as follows:

- `Interpolate`: This algorithm takes as input a set of  $t_c + 1$  pairs  $\{(x_i, y_i)\}_{i \in [t_c + 1]}$  of field elements where  $x_i \neq x_j$  for  $i \neq j$ . It outputs the unique polynomial  $C \in \mathbb{Z}_p[X]$  of degree at most  $t_c$  such that  $C(x_i) = y_i$  for all  $i \in [t_c + 1]$ . This can be done by computing the coefficients of the polynomial using standard Lagrange interpolation.
- `ExplInterpolate`: This algorithm takes as input a vector of  $t_r + 1$  group elements  $(S_1, \dots, S_{t_r + 1})$ . It outputs a vector of  $n + 1$  group elements  $T = (T_0, \dots, T_n)$  where  $T_j = \prod_{i \in [t_r + 1]} S_i^{L_{i,j}}$  for all  $j \in [n]$  and  $L_{i,j}$  denotes the  $i$ -th Lagrange coefficient for the set  $\{1, \dots, t_r + 1\}$  at the evaluation point  $j$ . Concretely, for all polynomials  $F \in \mathbb{Z}_p[X]$  of degree at most  $t_r$ , we have  $F(j) = \sum_{i=1}^{t_r + 1} L_{i,j} F(i)$ .

**Protocol Description.** As said, the formal description of HAVSS from the perspective of a party  $P_i$  is given in the full version. Conceptually, HAVSS has the following four steps:

1. **Dealer Committing Phase.** The dealer  $P_d$  samples a uniform bivariate polynomial  $S \in \mathbb{Z}_p[X, Y]$  of degree  $t_r$  in  $X$  and  $t_c$  in  $Y$  such that  $S(0, 0) = s$ . It then generates commitments  $\text{cm}_1, \dots, \text{cm}_{t_r + 1}$  to the (univariate) column polynomials  $C_1(Y) := S(1, Y), \dots, C_{t_r + 1}(Y) := S(t_r + 1, Y)$  of degree  $t_c$ . Concretely, these commitments are generalized Pedersen commitments and have the following form:

$$\text{cm}_i := \prod_{j=0}^{t_c} g_j^{c_j^i} \quad \text{where } C_i(Y) = \sum_{j=0}^{t_c} c_{j,i} Y^j \in \mathbb{Z}_p[Y].$$

Additionally, the dealer  $P_d$  computes for all  $i \in [t_r + 1]$  the exponentiated evaluations  $S_i := g^{S(i,0)}$  of the polynomial  $S(X, 0)$  and NIZK proofs  $\pi_i^{\text{exp}}$  for the relation  $\mathcal{R}_{\text{exp}}$ . Having done this, the dealer reliably broadcasts the message  $(\text{CM}, \text{row}_0)$  where  $\text{CM} = (\text{cm}_1, \dots, \text{cm}_{t_r + 1})$  are the commitments and  $\text{row}_0 := ((S_1, \pi_1^{\text{exp}}), \dots, (S_{t_r + 1}, \pi_{t_r + 1}^{\text{exp}}))$  are the exponentiated evaluations along with the NIZK proofs of correctness. Upon receiving this message, parties can compute commitments  $(\text{cm}_0, \dots, \text{cm}_n)$  to all column polynomials  $C_0(Y), \dots, C_n(Y)$  and the exponentiated evaluations  $S_i$  for all  $i \in [n]$  using `ExplInterpolate`. Here, we rely on the homomorphic properties of the commitments (cf. Remark 5).

2. **Dealer Distributing Rows.** The dealer proceeds by sending each party  $P_i$  the evaluations  $C_1(i), \dots, C_n(i)$  along the  $i$ -th row polynomial  $S(X, i)$ <sup>10</sup>. The

<sup>10</sup> Note that the identity  $C_j(i) = S(j, i)$  holds by definition of  $C_j$ .

dealer also sends for all  $j \in [n]$  proofs  $\pi_{j,i}$  for the relation  $\mathcal{R}_{\text{open}}$  attesting that the evaluation  $C_j(i)$  is correct with respect to the commitment  $\text{cm}_j$ . Upon receiving such a row along with the evaluation proofs from the dealer, each party  $P_i$  checks the correctness of the evaluations by verifying the proofs. Only in case all proofs verify, the party  $P_i$  distributes the row among all parties. This is done by sending to party  $P_j$  the evaluation  $C_j(i)$  along with the proof  $\pi_{j,i}$  in a “column” message. Additionally, it sends a “vote” message to all parties. Upon receiving  $t_c + 1$  “column” messages with valid proofs from other parties, a party  $P_i$  interpolates these received values to obtain a polynomial  $C_i(Y) \in \mathbb{Z}_p[Y]$  of degree  $t_c$ . This constitutes its column polynomial.

3. **Voting Phase.** Upon receiving “vote” messages from  $n - t_c$  parties, every party knows that at least  $n - 2t_c \geq t_c + 1$  honest parties received correct rows. These honest parties have evaluation points of other party’s column polynomials (one evaluation point per column polynomial) which they will forward to them. Therefore, every party will eventually receive enough points to interpolate its column polynomial and will be able to terminate. In order to signify that it thinks that all parties will be able to terminate, a party also sends a “done” message (upon receiving  $n - t_c$  vote messages).
4. **Termination.** Upon receiving “done” messages from  $t_c + 1$  parties, every party also sends a “done” message. Upon receiving “done” messages from  $n - t_c$  parties, and acquiring its polynomial  $C_i(Y)$  and the exponentiated evaluations  $S_0, \dots, S_n$  of the polynomial  $S(X, 0)$  of degree  $t_r$ , the party  $P_i$  terminates. This technique of echoing “done” messages is a Bracha-style termination gadget [22]. Before terminating, every party waits to receive  $n - t_c$  “done” messages. Out of those messages, at least  $n - 2t_c \geq t_c + 1$  were sent by honest parties. As a consequence, every party will receive at least  $t_c + 1$  “done” messages (those sent by honest parties) and thus send a “done” message as well. This guarantees eventual termination of all parties.
5. **Reconstruction.** During reconstruction, parties can simply output their shares  $s_i := C_i(0)$  and the vector  $S = (S_0, S_1, \dots, S_n)$  from the information collected during the sharing phase.

*Remark 5.* In our construction, we rely on the homomorphic properties of the Pedersen commitment in order to compute Pedersen commitments for the remaining column polynomials. Here, we sketch that this is possible using algorithm `ExplInterpolate`, i.e., Lagrange interpolation in the exponent. For the first  $t_r + 1$  column polynomials  $C_i(Y) = S(i, Y)$ ,  $i \in [t_r + 1]$ , we observe that  $C_i(y) = S(i, y) = \sum_{j=1}^{t_r+1} L_{i,j} S(j, y) = \sum_{j=1}^{t_r+1} L_{i,j} C_j(y)$  for all  $y \in \mathbb{Z}_p$  and all  $j \in [n]$ . Since this identity holds for all  $y \in \mathbb{Z}_p$ , it also has to hold as an identity of polynomials  $C_1, \dots, C_{t_r+1}$  in  $\mathbb{Z}_p[Y]$ . As a result, we obtain the equivalent identity  $S(i, Y) = \sum_{j=1}^{t_r+1} L_{i,j} S(j, Y)$  for the bivariate polynomial  $S(X, Y)$ . From this, we easily see that applying the same linear relation to the commitments of  $S(1, Y), \dots, S(t_r + 1, Y)$  (which are the first  $t_r + 1$  column polynomials  $C_1, \dots, C_{t_r+1}$ ) yields a commitment to  $S(i, Y)$  for any  $i \in [n]$ , as required.

## 6.2 Security Analysis

We proceed with the security analysis of our high-threshold AVSS scheme HAVSS (cf. Sect. 6.1). In the following, we give an intuition for the proof of oracle-aided simulatability. We omit the correctness and termination properties, since these follow from standard considerations.

**Proof Intuition.** The simulator  $\text{Sim}$  runs on an input of  $k := t_r + 1$  group elements  $\zeta := (\zeta_1, \dots, \zeta_k) \in \mathbb{G}^k$ . In order to simulate a sharing of a bivariate polynomial  $S(X, Y) \in \mathbb{Z}_p[X, Y]$  of degree  $t_r$  in  $X$  and  $t_c$  in  $Y$ , the simulator  $\text{Sim}$  embeds the given  $t_r + 1$  elements  $\zeta_1, \dots, \zeta_k$  into exponentiated evaluations of the polynomial  $S(X, 0)$  of degree  $t_r$  at the points  $\{1, \dots, t_r + 1\}$ . Since  $S(X, 0)$  is of degree  $t_r$ , these  $t_r + 1$  evaluations determine the remaining evaluations in the exponent (to base  $g$ ). By Lagrange interpolation in the exponent,  $\text{Sim}$  obtains evaluations of  $S(X, 0)$  in the exponent at all the points  $\{1, \dots, n\}$ . Next, it samples  $t_r + 1$  commitments  $\text{cm}_1, \dots, \text{cm}_{t_r+1} \leftarrow_{\$} \mathbb{G}$  to the first  $t_r + 1$  column polynomials  $C_i(Y) := S(i, Y)$  uniformly at random, and interpolates them in the exponent to obtain the commitments  $\text{cm}_1, \dots, \text{cm}_n$  to all column polynomials. From this point on, while simulating we make sure that parties' messages are consistent with the commitments and with the polynomial  $S(X, 0)$ . This mainly involves sending messages normally while carefully generating a corrupted party  $P_i$ 's view upon corruption. This is done by calling the discrete logarithm oracle (which is provided to  $\text{Sim}$  by definition of oracle-aided simulatability) on input element  $S_i := g^{S(i,0)}$  to obtain  $S(i, 0)$ , and sampling polynomials for  $P_i$  that is consistent with these  $S(i, 0)$  and with the previously defined polynomials for all other corrupted parties. In this way, the simulator  $\text{Sim}$  makes at most  $t_c = k - \delta_a$  calls to the discrete logarithm oracle which is the correct number of total calls according to our definition of oracle-aided simulatability. All opening proofs, along with the exponentiated opening proofs for the  $S_i$  elements can be produced by simulating the NIZKs for the relations  $\mathcal{R}_{\text{open}}$  and  $\mathcal{R}_{\text{exp}}$ . We provide a full proof of the following theorem in the full version.

**Theorem 3 (AVSS).** *Let  $t_c, t_r, n \in \mathbb{N}$  be natural numbers such that  $t_c < n/3$  and  $t_r \in [t_c, n - t_c]$ . Further, let  $\text{PS}_{\text{open}}$  and  $\text{PS}_{\text{exp}}$  be zero-knowledge proofs of knowledge and let the DLOG assumption hold relative to  $(\mathbb{G}, p, g)$ . Then, assuming secure erasures, HAVSS (see full version) is an oracle-aided secure  $(t_c, t_r, n)$ -threshold AVSS scheme.*

*Remark 6.* We note that secure erasures are only used in the protocol to erase the randomness for generating the proofs. The reason for this is as follows: in our simulation, we simulate the proofs using the zero-knowledge property. Upon an adaptive corruption, we would have to provide the randomness used for generating the proofs if the protocol did not specify erasing it. Hence, if the underlying proof system is explainable as defined in [59], we would not need to rely on erasures for our construction. An example of an explainable proof is the Schnorr NIZK proof, where we can compute the randomness  $r$  from the witness  $w$  and a simulated proof  $\sigma$  as  $r := \sigma - c \cdot w$ .

### 6.3 Comparison to Other High-Threshold AVSS

In the following, we highlight how our high-threshold AVSS construction differs from previous ones. In the full version, we elaborate on the other high-threshold AVSS schemes [2, 5, 44, 64]. Our high-threshold AVSS is closest to Bingo [2] and HAVEN [5]. On a high level, we augment HAVEN with a bivariate polynomial structure and give a more modular and cleaner design. The bivariate structure allows us to give an adaptive security proof as in [2]. Also, we can remove the need of tester polynomials and the vector commitment VC used in HAVEN. Further, we note that augmenting HAVEN with a bivariate structure while making everything consistent is not trivial. In particular, the adaptive security proof is highly challenging as evident in previous works [2, 8, 38]. Finally, by letting the dealer commit to column polynomials and distribute row polynomials, we avoid the need of homomorphic proofs, pairings, and the KZG-setup as required in Bingo. We also refer to Table 2 for a comparison of representative AVSS schemes.

## 7 Instantiation and Efficiency

In this section, we instantiate our framework with concrete building blocks to obtain HARTS and evaluate its communication and round complexity.

**Instantiation.** For an overview of our instantiation, we refer to Fig. 1. Concretely, we use an upper-triangular Pascal matrix [57] for the superinvertible matrix. Further, we use VABA [2] for the MVBA protocol. Finally, we use our HAVSS (cf. Sect. 6) for the AVSS scheme with the following specifications: the protocol from [43] for the reliable broadcast, and Bulletproofs [24] for the inner product arguments. Further, we note that online-extractability of Bulletproofs has been studied before [51, 56]. We emphasize that other inner product arguments could also be used in our framework, in particular modern versions of Bulletproofs [33, 48] to improve the concrete efficiency. Our choice of Bulletproofs is based on the fact that it is most widely used in practical applications to date. Finally, we note that KZG proofs [62] are not suitable for our instantiation: first, KZG relies on a trusted setup and a long structured common reference string; second, and more importantly, KZG relies on pairings. Our objective is to obtain a Schnorr-compatible threshold signature, and thus we cannot assume to have access to a pairing, as Schnorr is typically implemented over pairing-free groups.

**Efficiency.** We evaluate the communication and round complexity of our threshold Schnorr signature scheme HARTS. In our AVSS scheme HAVSS, the dealer reliably broadcasts a vector of Pedersen commitments of size  $O(\lambda n)$  along with  $O(n)$  group elements and proofs of correctness. Using the reliable broadcast protocol from [43] and Bulletproofs [24], this step has a communication cost of  $O(\lambda n^2 \log n)$ . Further, the dealer privately sends  $n$  field elements and evaluation proofs to each party, who then disperses these values among all parties. This step also has a communication cost of  $O(\lambda n^2 \log n)$ . Thus, we see that HAVSS has log-quadratic communication cost. As each party invokes it once, the overall cost

of the AVSS sharing phase is log-cubic. Next, the MVBA protocol VABA [2] has cubic communication cost and terminates in expected constant rounds. Obviously, the application of the matrix SI thereafter is only local and does not affect the communication and round complexity of the protocol.

From this analysis, we see that the protocol PADKG (see the full version for a formal protocol description) for nonce generation generates  $\ell = t_c + 1 \in O(n)$  nonces with a communication complexity of  $O(\lambda n^3 \log n)$ . Thus, we obtain an amortized communication cost of  $O(\lambda n^2 \log n)$  per nonce. Finally, for signature generation, each party sends a threshold Schnorr signature share of size  $O(\lambda)$  to all other parties. Since this step has a communication cost of  $O(\lambda n^2)$ , we find that a total of  $O(\lambda n^3 \log n)$  communication is required to generate  $O(n)$  signatures in expected constant rounds, as desired. We note that PADKG generates  $\ell \in O(n)$  nonces in expected constant rounds, but only a single round is needed after that to sign a message. In particular, our security analysis allows parties to run any polynomially-bounded number of instances of PADKG offline (i.e., in the background or as precomputation), whose generated nonces can then be consumed individually (or in batches) in a single-round online phase upon a signing request.

**Acknowledgments.** CISPA authors are funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 507237585, and by the European Union, ERC-StG-2023-101116713. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Gilad Stern was Supported in part by ISF 2338/23, AFOSR Award FA9550-23-1-0387, AFOSR Award FA9550-23-1-0312, and an Algorand Foundation grant. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Government, AFOSR or the Algorand Foundation.

## References

1. Abe, M., Fehr, S.: Adaptively secure feldman VSS and applications to universally-composable threshold cryptography. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 317–334. Springer, Heidelberg (Aug 2004). [https://doi.org/10.1007/978-3-540-28628-8\\_20](https://doi.org/10.1007/978-3-540-28628-8_20)
2. Abraham, I., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G.: Bingo: Adaptivity and asynchrony in verifiable secret sharing and distributed key generation. In: Advances in Cryptology - CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I. p. 39-70. Springer-Verlag, Berlin, Heidelberg (2023). [https://doi.org/10.1007/978-3-031-38557-5\\_2](https://doi.org/10.1007/978-3-031-38557-5_2), [https://doi.org/10.1007/978-3-031-38557-5\\_2](https://doi.org/10.1007/978-3-031-38557-5_2)
3. Abraham, I., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G., Tomescu, A.: Reaching consensus for asynchronous distributed key generation. In: 40th ACM Symposium Annual on Principles of Distributed Computing. pp. 363–373. Association for Computing Machinery, Portland, OR, USA (2021)

4. Alhaddad, N., Varia, M., Yang, Z.: Haven++: Batched and packed dual-threshold asynchronous complete secret sharing with applications. *Cryptology ePrint Archive*, Paper 2024/326 (2024), <https://eprint.iacr.org/2024/326>, <https://eprint.iacr.org/2024/326>
5. Alhaddad, N., Varia, M., Zhang, H.: High-threshold AVSS with optimal communication complexity. In: Borisov, N., Díaz, C. (eds.) *FC 2021, Part II*. LNCS, vol. 12675, pp. 479–498. Springer, Heidelberg (Mar 2021). [https://doi.org/10.1007/978-3-662-64331-0\\_25](https://doi.org/10.1007/978-3-662-64331-0_25)
6. Almansa, J.F., Damgård, I., Nielsen, J.B.: Simplified threshold RSA with adaptive and proactive security. In: Vaudenay, S. (ed.) *EUROCRYPT 2006*. LNCS, vol. 4004, pp. 593–611. Springer, Heidelberg (May / Jun 2006). [https://doi.org/10.1007/11761679\\_35](https://doi.org/10.1007/11761679_35)
7. Bacho, R., Lenzen, C., Loss, J., Ochsenreither, S., Papachristoudis, D.: Grandline: Adaptively secure dkg and randomness beacon with (almost) quadratic communication complexity. *Cryptology ePrint Archive*, Paper 2023/1887 (2023), <https://eprint.iacr.org/2023/1887>, <https://eprint.iacr.org/2023/1887>
8. Bacho, R., Loss, J.: On the adaptive security of the threshold BLS signature scheme. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) *ACM CCS 2022*. pp. 193–207. ACM Press (Nov 2022). <https://doi.org/10.1145/3548606.3560656>
9. Bacho, R., Loss, J., Stern, G., Wagner, B.: HARTS: High-threshold, adaptively secure, and robust threshold schnorr signatures. *Cryptology ePrint Archive*, Paper 2024/280 (2024), <https://eprint.iacr.org/2024/280>
10. Bacho, R., Loss, J., Tessaro, S., Wagner, B., Zhu, C.: Twinkle: Threshold signatures from ddh with full adaptive security. *Cryptology ePrint Archive*, Paper 2023/1482 (2023), <https://eprint.iacr.org/2023/1482>, <https://eprint.iacr.org/2023/1482>
11. Backes, M., Datta, A., Kate, A.: Asynchronous computational VSS with reduced communication complexity. In: Dawson, E. (ed.) *CT-RSA 2013*. LNCS, vol. 7779, pp. 259–276. Springer, Heidelberg (Feb / Mar 2013). [https://doi.org/10.1007/978-3-642-36095-4\\_17](https://doi.org/10.1007/978-3-642-36095-4_17)
12. Baldimtsi, F., Chalkias, K.K., Garillot, F., Lindstrom, J., Riva, B., Roy, A., Sedaghat, M., Sonnino, A., Waiwitlikhit, P., Wang, J.: Subset-optimized bls multi-signature with key aggregation. *Cryptology ePrint Archive*, Paper 2023/498 (2023), <https://eprint.iacr.org/2023/498>, <https://eprint.iacr.org/2023/498>
13. Beaver, D., Haber, S.: Cryptographic protocols provably secure against dynamic adversaries. In: Rueppel, R.A. (ed.) *EUROCRYPT’92*. LNCS, vol. 658, pp. 307–323. Springer, Heidelberg (May 1993). [https://doi.org/10.1007/3-540-47555-9\\_26](https://doi.org/10.1007/3-540-47555-9_26)
14. Bellare, M., Crites, E.C., Komlo, C., Maller, M., Tessaro, S., Zhu, C.: Better than advertised security for non-interactive threshold signatures. In: Dodis, Y., Shrimpton, T. (eds.) *CRYPTO 2022, Part IV*. LNCS, vol. 13510, pp. 517–550. Springer, Heidelberg (Aug 2022). [https://doi.org/10.1007/978-3-031-15985-5\\_18](https://doi.org/10.1007/978-3-031-15985-5_18)
15. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The one-more-RSA-inversion problems and the security of Chaum’s blind signature scheme. *Journal of Cryptology* **16**(3), 185–215 (Jun 2003). <https://doi.org/10.1007/s00145-002-0120-1>
16. Bellare, M., Tessaro, S., Zhu, C.: Stronger security for non-interactive threshold signatures: BLS and FROST. *Cryptology ePrint Archive*, Report 2022/833 (2022), <https://eprint.iacr.org/2022/833>
17. Ben-Or, M., Canetti, R., Goldreich, O.: Asynchronous secure computation. In: 25th ACM STOC. pp. 52–61. ACM Press (May 1993). <https://doi.org/10.1145/167088.167109>

18. Benhamouda, F., Halevi, S., Krawczyk, H., Ma, Y., Rabin, T.: Sprint: High-throughput robust distributed schnorr signatures. In: Joye, M., Leander, G. (eds.) *Advances in Cryptology – EUROCRYPT 2024*. pp. 62–91. Springer Nature Switzerland, Cham (2024)
19. Benhamouda, F., Lepoint, T., Loss, J., Orrù, M., Raykova, M.: On the (in)security of ROS. In: Canteaut, A., Standaert, F.X. (eds.) *EUROCRYPT 2021, Part I*. LNCS, vol. 12696, pp. 33–53. Springer, Heidelberg (Oct 2021). [https://doi.org/10.1007/978-3-030-77870-5\\_2](https://doi.org/10.1007/978-3-030-77870-5_2)
20. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: 20th ACM STOC. pp. 103–112. ACM Press (May 1988). <https://doi.org/10.1145/62212.62222>
21. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y. (ed.) *PKC 2003*. LNCS, vol. 2567, pp. 31–46. Springer, Heidelberg (Jan 2003). [https://doi.org/10.1007/3-540-36288-6\\_3](https://doi.org/10.1007/3-540-36288-6_3)
22. Bracha, G.: An asynchronous  $[(n - 1)/3]$ -resilient consensus protocol. In: *Proceedings of the Third Annual ACM Symposium on Principles of Distributed Computing*. p. 154-162. PODC '84, Association for Computing Machinery, New York, NY, USA (1984). <https://doi.org/10.1145/800222.806743>, <https://doi.org/10.1145/800222.806743>
23. Brandão, L.T.A.N., Peralta, R.: Nist first call for multi-party threshold schemes (2023), <https://csrc.nist.gov/pubs/ir/8214/c/ipd>, nIST IR 8214C (Initial Public Draft)
24. Bünz, B., Bootle, J., Boneh, D., Poelstra, A., Wuille, P., Maxwell, G.: Bulletproofs: Short proofs for confidential transactions and more. In: 2018 IEEE Symposium on Security and Privacy. pp. 315–334. IEEE Computer Society Press (May 2018). <https://doi.org/10.1109/SP.2018.00020>
25. Cachin, C., Kursawe, K., Lysyanskaya, A., Strohli, R.: Asynchronous verifiable secret sharing and proactive cryptosystems. In: Atluri, V. (ed.) *ACM CCS 2002*. pp. 88–97. ACM Press (Nov 2002). <https://doi.org/10.1145/586110.586124>
26. Cachin, C., Kursawe, K., Petzold, F., Shoup, V.: Secure and efficient asynchronous broadcast protocols. In: Kilian, J. (ed.) *CRYPTO 2001*. LNCS, vol. 2139, pp. 524–541. Springer, Heidelberg (Aug 2001). [https://doi.org/10.1007/3-540-44647-8\\_31](https://doi.org/10.1007/3-540-44647-8_31)
27. Canetti, R.: Security and composition of multiparty cryptographic protocols. *Journal of Cryptology* **13**(1), 143–202 (Jan 2000). <https://doi.org/10.1007/s001459910006>
28. Canetti, R., Gennaro, R., Goldfeder, S., Makriyannis, N., Peled, U.: UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) *ACM CCS 2020*. pp. 1769–1787. ACM Press (Nov 2020). <https://doi.org/10.1145/3372297.3423367>
29. Canetti, R., Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Adaptive security for threshold cryptosystems. In: Wiener, M.J. (ed.) *CRYPTO'99*. LNCS, vol. 1666, pp. 98–115. Springer, Heidelberg (Aug 1999). [https://doi.org/10.1007/3-540-48405-1\\_7](https://doi.org/10.1007/3-540-48405-1_7)
30. Canetti, R., Rabin, T.: Fast asynchronous byzantine agreement with optimal resilience. In: 25th ACM STOC. pp. 42–51. ACM Press (May 1993). <https://doi.org/10.1145/167088.167105>
31. Choudhury, A., Patra, A.: An efficient framework for unconditionally secure multiparty computation. *IEEE Trans. Inf. Theor.* **63**(1), 428-468 (jan 2017). <https://doi.org/10.1109/TIT.2016.2614685>, <https://doi.org/10.1109/TIT.2016.2614685>



32. Chu, H., Gerhart, P., Ruffing, T., Schröder, D.: Practical schnorr threshold signatures without the algebraic group model. In: *Advances in Cryptology - CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*. p. 743-773. Springer-Verlag, Berlin, Heidelberg (2023). [https://doi.org/10.1007/978-3-031-38557-5\\_24](https://doi.org/10.1007/978-3-031-38557-5_24), [https://doi.org/10.1007/978-3-031-38557-5\\_24](https://doi.org/10.1007/978-3-031-38557-5_24)
33. Chung, H., Han, K., Ju, C., Kim, M., Seo, J.H.: Bulletproofs+: Shorter proofs for a privacy-enhanced distributed ledger. *IEEE Access* **10**, 42067–42082 (2022), <https://api.semanticscholar.org/CorpusID:220118175>
34. Cohen, R., shelat, a., Wichs, D.: Adaptively secure MPC with sublinear communication complexity. In: Boldyreva, A., Micciancio, D. (eds.) *CRYPTO 2019, Part II*. LNCS, vol. 11693, pp. 30–60. Springer, Heidelberg (Aug 2019). [https://doi.org/10.1007/978-3-030-26951-7\\_2](https://doi.org/10.1007/978-3-030-26951-7_2)
35. Cramer, R., Damgård, I., Dziembowski, S., Hirt, M., Rabin, T.: Efficient multiparty computations secure against an adaptive adversary. In: Stern, J. (ed.) *EUROCRYPT'99*. LNCS, vol. 1592, pp. 311–326. Springer, Heidelberg (May 1999). [https://doi.org/10.1007/3-540-48910-X\\_22](https://doi.org/10.1007/3-540-48910-X_22)
36. Crites, E., Kohlweiss, M., Preneel, B., Sedaghat, M., Slamanig, D.: Threshold structure-preserving signatures. In: *Advances in Cryptology - ASIACRYPT 2023: 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part II*. p. 348-382. Springer-Verlag, Berlin, Heidelberg (2023). [https://doi.org/10.1007/978-981-99-8724-5\\_11](https://doi.org/10.1007/978-981-99-8724-5_11), [https://doi.org/10.1007/978-981-99-8724-5\\_11](https://doi.org/10.1007/978-981-99-8724-5_11)
37. Crites, E., Komlo, C., Maller, M.: How to prove schnorr assuming schnorr: Security of multi- and threshold signatures. *Cryptology ePrint Archive, Report 2021/1375* (2021), <https://eprint.iacr.org/2021/1375>
38. Crites, E., Komlo, C., Maller, M.: Fully adaptive schnorr threshold signatures. In: *Advances in Cryptology - CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*. p. 678-709. Springer-Verlag, Berlin, Heidelberg (2023). [https://doi.org/10.1007/978-3-031-38557-5\\_22](https://doi.org/10.1007/978-3-031-38557-5_22), [https://doi.org/10.1007/978-3-031-38557-5\\_22](https://doi.org/10.1007/978-3-031-38557-5_22)
39. Crites, E., Komlo, C., Maller, M., Tessaro, S., Zhu, C.: Snowblind: A threshold blind signature in pairing-free groups. In: *Advances in Cryptology - CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*. p. 710-742. Springer-Verlag, Berlin, Heidelberg (2023). [https://doi.org/10.1007/978-3-031-38557-5\\_23](https://doi.org/10.1007/978-3-031-38557-5_23), [https://doi.org/10.1007/978-3-031-38557-5\\_23](https://doi.org/10.1007/978-3-031-38557-5_23)
40. Dalskov, A.P.K., Orlandi, C., Keller, M., Shrishak, K., Shulman, H.: Securing DNSSEC keys via threshold ECDSA from generic MPC. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) *ESORICS 2020, Part II*. LNCS, vol. 12309, pp. 654–673. Springer, Heidelberg (Sep 2020). [https://doi.org/10.1007/978-3-030-59013-0\\_32](https://doi.org/10.1007/978-3-030-59013-0_32)
41. Das, S., Ren, L.: Adaptively secure bls threshold signatures from ddh and co-cdh. *Cryptology ePrint Archive, Paper 2023/1553* (2023), <https://eprint.iacr.org/2023/1553>, <https://eprint.iacr.org/2023/1553>
42. Das, S., Xiang, Z., Kokoris-Kogias, L., Ren, L.: Practical asynchronous high-threshold distributed key generation and distributed polynomial sampling. In: *32nd USENIX Security Symposium (USENIX Security 23)*. pp. 5359–5376. USENIX Association, Anaheim, CA (Aug 2023), <https://www.usenix.org/conference/usenixsecurity23/presentation/das>

43. Das, S., Xiang, Z., Ren, L.: Balanced quadratic reliable broadcast and improved asynchronous verifiable information dispersal. *Cryptology ePrint Archive*, Report 2022/052 (2022), <https://eprint.iacr.org/2022/052>
44. Das, S., Yurek, T., Xiang, Z., Miller, A.K., Kokoris-Kogias, L., Ren, L.: Practical asynchronous distributed key generation. In: 2022 IEEE Symposium on Security and Privacy. pp. 2518–2534. IEEE Computer Society Press (May 2022). <https://doi.org/10.1109/SP46214.2022.9833584>
45. Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Pomerance, C. (ed.) CRYPTO'87. LNCS, vol. 293, pp. 120–127. Springer, Heidelberg (Aug 1988). [https://doi.org/10.1007/3-540-48184-2\\_8](https://doi.org/10.1007/3-540-48184-2_8)
46. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 307–315. Springer, Heidelberg (Aug 1990). [https://doi.org/10.1007/0-387-34805-0\\_28](https://doi.org/10.1007/0-387-34805-0_28)
47. Drijvers, M., Edalatnejad, K., Ford, B., Kiltz, E., Loss, J., Neven, G., Stepanovs, I.: On the security of two-round multi-signatures. In: 2019 IEEE Symposium on Security and Privacy. pp. 1084–1101. IEEE Computer Society Press (May 2019). <https://doi.org/10.1109/SP.2019.00050>
48. Eagen, L., Kanjalkar, S., Ruffing, T., Nick, J.: Bulletproofs++: Next generation confidential transactions via reciprocal set membership arguments. In: Joye, M., Leander, G. (eds.) *Advances in Cryptology – EUROCRYPT 2024*. pp. 249–279. Springer Nature Switzerland, Cham (2024)
49. Frankel, Y., MacKenzie, P.D., Yung, M.: Adaptively-secure optimal-resilience proactive RSA. In: Lam, K.Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT'99. LNCS, vol. 1716, pp. 180–194. Springer, Heidelberg (Nov 1999). [https://doi.org/10.1007/978-3-540-48000-6\\_15](https://doi.org/10.1007/978-3-540-48000-6_15)
50. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Heidelberg (Aug 2018). [https://doi.org/10.1007/978-3-319-96881-0\\_2](https://doi.org/10.1007/978-3-319-96881-0_2)
51. Ganesh, C., Orlandi, C., Pancholi, M., Takahashi, A., Tschudi, D.: Fiat-shamir bulletproofs are non-malleable (in the algebraic group model). In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 397–426. Springer, Heidelberg (May / Jun 2022). [https://doi.org/10.1007/978-3-031-07085-3\\_14](https://doi.org/10.1007/978-3-031-07085-3_14)
52. Gelashvili, R., Kokoris-Kogias, L., Sonnino, A., Spiegelman, A., Xiang, Z.: Jolteon and ditto: Network-adaptive efficient consensus with asynchronous fallback. In: Eyal, I., Garay, J.A. (eds.) FC 2022. LNCS, vol. 13411, pp. 296–315. Springer, Heidelberg (May 2022). [https://doi.org/10.1007/978-3-031-18283-9\\_14](https://doi.org/10.1007/978-3-031-18283-9_14)
53. Gennaro, R., Goldfeder, S.: Fast multiparty threshold ECDSA with fast trustless setup. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 1179–1194. ACM Press (Oct 2018). <https://doi.org/10.1145/3243734.3243859>
54. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 295–310. Springer, Heidelberg (May 1999). [https://doi.org/10.1007/3-540-48910-X\\_21](https://doi.org/10.1007/3-540-48910-X_21)
55. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. *Journal of Cryptology* **20**(1), 51–83 (Jan 2007). <https://doi.org/10.1007/s00145-006-0347-3>

56. Ghoshal, A., Tessaro, S.: Tight state-restoration soundness in the algebraic group model. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 64–93. Springer, Heidelberg, Virtual Event (Aug 2021). [https://doi.org/10.1007/978-3-030-84252-9\\_3](https://doi.org/10.1007/978-3-030-84252-9_3)
57. Groth, J., Shoup, V.: Fast batched asynchronous distributed key generation. In: Advances in Cryptology - EUROCRYPT 2024: 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26–30, 2024, Proceedings, Part V. p. 370–400. Springer-Verlag, Berlin, Heidelberg (2024). [https://doi.org/10.1007/978-3-031-58740-5\\_13](https://doi.org/10.1007/978-3-031-58740-5_13), [https://doi.org/10.1007/978-3-031-58740-5\\_13](https://doi.org/10.1007/978-3-031-58740-5_13)
58. Gurkan, K., Jovanovic, P., Maller, M., Meiklejohn, S., Stern, G., Tomescu, A.: Aggregatable distributed key generation. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 147–176. Springer, Heidelberg (Oct 2021). [https://doi.org/10.1007/978-3-030-77870-5\\_6](https://doi.org/10.1007/978-3-030-77870-5_6)
59. Hanzlik, L., Kluczniak, K.: Explainable arguments. In: Eyal, I., Garay, J.A. (eds.) FC 2022. LNCS, vol. 13411, pp. 59–79. Springer, Heidelberg (May 2022). [https://doi.org/10.1007/978-3-031-18283-9\\_4](https://doi.org/10.1007/978-3-031-18283-9_4)
60. Hirt, M., Nielsen, J.B.: Robust multiparty computation with linear communication complexity. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 463–482. Springer, Heidelberg (Aug 2006). [https://doi.org/10.1007/11818175\\_28](https://doi.org/10.1007/11818175_28)
61. Jarecki, S., Lysyanskaya, A.: Adaptively secure threshold cryptography: Introducing concurrency, removing erasures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 221–242. Springer, Heidelberg (May 2000). [https://doi.org/10.1007/3-540-45539-6\\_16](https://doi.org/10.1007/3-540-45539-6_16)
62. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 177–194. Springer, Heidelberg (Dec 2010). [https://doi.org/10.1007/978-3-642-17373-8\\_11](https://doi.org/10.1007/978-3-642-17373-8_11)
63. Katz, J., Yung, M.: Threshold cryptosystems based on factoring. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 192–205. Springer, Heidelberg (Dec 2002). [https://doi.org/10.1007/3-540-36178-2\\_12](https://doi.org/10.1007/3-540-36178-2_12)
64. Kokoris-Kogias, E., Malkhi, D., Spiegelman, A.: Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1751–1767. ACM Press (Nov 2020). <https://doi.org/10.1145/3372297.3423364>
65. Komlo, C., Goldberg, I.: FROST: Flexible round-optimized Schnorr threshold signatures. In: Dunkelman, O., Jr., M.J.J., O’Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 34–65. Springer, Heidelberg (Oct 2020). [https://doi.org/10.1007/978-3-030-81652-0\\_2](https://doi.org/10.1007/978-3-030-81652-0_2)
66. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. 4(3), 382–401 (jul 1982). <https://doi.org/10.1145/357172.357176>, <https://doi.org/10.1145/357172.357176>
67. Libert, B., Joye, M., Yung, M.: Born and raised distributively: fully distributed non-interactive adaptively-secure threshold signatures with short shares. In: Halldórsson, M.M., Dolev, S. (eds.) 33rd ACM PODC. pp. 303–312. ACM (Jul 2014). <https://doi.org/10.1145/2611462.2611498>
68. Lindell, Y.: Simple three-round multiparty schnorr signing with full simulatability. Cryptology ePrint Archive, Paper 2022/374 (2022), <https://eprint.iacr.org/2022/374>, <https://eprint.iacr.org/2022/374>

69. Lysyanskaya, A., Peikert, C.: Adaptive security in the threshold setting: From cryptosystems to signature schemes. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 331–350. Springer, Heidelberg (Dec 2001). [https://doi.org/10.1007/3-540-45682-1\\_20](https://doi.org/10.1007/3-540-45682-1_20)
70. Nick, J., Ruffing, T., Seurin, Y.: MuSig2: Simple two-round Schnorr multi-signatures. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 189–221. Springer, Heidelberg, Virtual Event (Aug 2021). [https://doi.org/10.1007/978-3-030-84242-0\\_8](https://doi.org/10.1007/978-3-030-84242-0_8)
71. Nielsen, J.B.: Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In: Yung, M. (ed.) Advances in Cryptology — CRYPTO 2002. pp. 111–126. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
72. Nikolaenko, V., Ragsdale, S., Bonneau, J., Boneh, D.: Powers-of-tau to the people: Decentralizing setup ceremonies. In: Pöpper, C., Batina, L. (eds.) Applied Cryptography and Network Security. pp. 105–134. Springer Nature Switzerland, Cham (2024)
73. Ruffing, T., Ronge, V., Jin, E., Schneider-Bensch, J., Schröder, D.: ROAST: Robust asynchronous schnorr threshold signatures. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 2551–2564. ACM Press (Nov 2022). <https://doi.org/10.1145/3548606.3560583>
74. Schnorr, C.P.: Efficient signature generation by smart cards. *Journal of Cryptology* 4(3), 161–174 (Jan 1991). <https://doi.org/10.1007/BF00196725>
75. Sedghighadikolaei, K., Yavuz, A.A.: A comprehensive survey of threshold digital signatures: Nist standards, post-quantum cryptography, exotic techniques, and real-world applications (2023)
76. Shoup, V.: Practical threshold signatures. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 207–220. Springer, Heidelberg (May 2000). [https://doi.org/10.1007/3-540-45539-6\\_15](https://doi.org/10.1007/3-540-45539-6_15)
77. Shoup, V.: The many faces of schnorr. *Cryptology ePrint Archive*, Paper 2023/1019 (2023), <https://eprint.iacr.org/2023/1019>, <https://eprint.iacr.org/2023/1019>
78. Shoup, V., Smart, N.P.: Lightweight asynchronous verifiable secret sharing with optimal resilience. *Cryptology ePrint Archive*, Paper 2023/536 (2023), <https://eprint.iacr.org/2023/536>, <https://eprint.iacr.org/2023/536>
79. Shrestha, N., Bhat, A., Kate, A., Nayak, K.: Synchronous distributed key generation without broadcasts. *Cryptology ePrint Archive*, Report 2021/1635 (2021), <https://eprint.iacr.org/2021/1635>
80. Stinson, D.R., Strobl, R.: Provably secure distributed Schnorr signatures and a  $(t, n)$  threshold scheme for implicit certificates. In: Varadharajan, V., Mu, Y. (eds.) ACISP 01. LNCS, vol. 2119, pp. 417–434. Springer, Heidelberg (Jul 2001). [https://doi.org/10.1007/3-540-47719-5\\_33](https://doi.org/10.1007/3-540-47719-5_33)
81. Tessaro, S., Zhu, C.: Threshold and multi-signature schemes from linear hash functions. In: *Advances in Cryptology - EUROCRYPT 2023: 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Lyon, France, April 23–27, 2023, Proceedings, Part V. p. 628–658. Springer-Verlag, Berlin, Heidelberg (2023). [https://doi.org/10.1007/978-3-031-30589-4\\_22](https://doi.org/10.1007/978-3-031-30589-4_22), [https://doi.org/10.1007/978-3-031-30589-4\\_22](https://doi.org/10.1007/978-3-031-30589-4_22)
82. Wuille, P., Nick, J., Ruffing, T.: Schnorr signatures for secp256k1. bitcoin improvement proposal 340. Github (Jan 2020), <https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki>

83. Yin, M., Malkhi, D., Reiter, M.K., Golan-Gueta, G., Abraham, I.: HotStuff: BFT consensus with linearity and responsiveness. In: Robinson, P., Ellen, F. (eds.) 38th ACM PODC. pp. 347–356. ACM (Jul / Aug 2019). <https://doi.org/10.1145/3293611.3331591>
84. Yurek, T., Luo, L., Fairoze, J., Kate, A., Miller, A.: hbacss: How to robustly share many secrets. Proceedings of the Network and Distributed System Security Symposium (NDSS) 2022 (01 2022). <https://doi.org/10.14722/ndss.2022.23120>



# Tiresias: Large Scale, UC-Secure Threshold Paillier

Offir Friedman<sup>(✉)</sup>, Avichai Marmor, Dolev Mutzari, Yehonatan C. Scaly,  
Yuval Spiizer, and Avishay Yanai

dWallet Labs, Tel Aviv, Israel  
research@dwalletlabs.com

**Abstract.** In the threshold version of Paillier’s encryption scheme, a set of parties collectively holds the secret decryption key through a secret sharing scheme. Whenever a ciphertext is to be decrypted, the parties send their decryption shares, which are then verified for correctness and combined into the plaintext. The scheme has been widely adopted in various applications, from secure voting to general purpose MPC protocols. However, among the handful of existing proposals for a maliciously secure scheme, one must choose between an efficient implementation that relies on non-standard assumptions or a computationally expensive implementation that relies on widely acceptable assumptions.

In this work, we show that one can enjoy the benefits of both worlds. Specifically, we adjust a scheme by Damgård et al. (Int. J. Inf. Secur. 2010) to get a practical distributed key generation (DKG). While the original scheme was only known to be secure under ad-hoc non-standard assumptions, we prove that the adjusted scheme is in fact secure under the decisional composite residuosity (DCR) assumption alone, required for the semantic security of the Paillier encryption scheme itself. This is possible thanks to a *novel reduction technique*, from *computing* and *proving* a false decryption share, to the factoring problem. Specifically, while there may exist false decryption shares for which the zk-proof verifies with non-negligible probability, they are computationally hard to find. Furthermore, we use similar ideas to prove that batching techniques by Aditya et al. (ACNS 2004), which allows a prover to *batch* several statements into a single proof, can be applied to our adjusted scheme. This enables a batched threshold Paillier decryption in the fully distributed setting for the first time.

Until now, verifying that a decryption share is correct was the bottleneck of threshold Paillier schemes and hindered real world deployments (unless one is willing to rely on a trusted dealer). Our work accumulates to shifting the bottleneck back to the plaintext reconstruction, just like in the semi-honest setting, and renders threshold Paillier practical for the first time, supporting large scale deployments.

We exemplify this shift by implementing the scheme and report our evaluation with up to 1000 parties, in the dishonest majority setting. Over

---

This research was conducted by the Cryptography Research team at dWallet Labs, as part of the research and development of the Odsy Network, a decentralized and universal access control layer launched by the Odsy Foundation.

For the full and most up-to-date version of this work, see [FMM+23].

an EC2 c6i machine, we get a throughput of about 50 and 3.6 decryptions per second, when run over a network of 100 and 1000 parties, respectively.

**Keywords:** Additive Homomorphic Encryption · Paillier Encryption · Threshold Encryption · Batched ZK Arguments

## 1 Introduction

The Paillier encryption scheme from 1999 [Pai99] has gained significant popularity due to its advantageous properties. It is a public-key encryption scheme renowned for its additive homomorphic property, enabling linear operations on encrypted data without requiring decryption. Additionally, the Paillier scheme supports a large message space, enabling useful operations on secrets.

Motivated by applications in voting systems, several authors have proposed *threshold* variants of the Paillier encryption scheme [FPS01, DJN10]. These variants are based on similar constructions for RSA signatures [Sho00].

A threshold encryption scheme facilitates a set of parties utilizing a public encryption key  $pk$  to encrypt messages while *collectively maintaining* the corresponding secret key  $sk$  for decrypting ciphertexts. In this scheme, each party  $\mathcal{P}_j$  possesses a secret decryption key share  $sk_j$ . When the parties collectively decide to decrypt a ciphertext  $ct$ , they participate in a cryptographic protocol that ultimately reveals the message while ensuring the confidentiality of the secret decryption key. Typically in such protocols, each party  $P_j$  broadcasts a “decryption share”  $ct_j = Dec_{sk_j}(ct)$ . If a sufficient number of parties, passing a certain pre-defined threshold, broadcast their decryption shares, those can then be locally combined by anyone to recover the plaintext  $pt = Dec_{sk}(ct)$ .

The combination of homomorphic properties and threshold decryption capabilities has rendered the Paillier encryption scheme highly appealing in systems focused on privacy-preserving voting [KLM+20, DJN10] and data aggregation in general [MT21, BS21]. Moreover, threshold decryption of the Paillier scheme (and additively homomorphic encryption in general) serves as a foundational building block in other cryptographic protocols like threshold signatures [GGN16] and secure multiparty computation (MPC) in general [DN03, DPSZ12]. In many prior works, such as [FPS01, GGN16], the focus was primarily on the threshold Paillier decryption feature. However, these approaches often relied on a trusted dealer for key generation and secret key share distribution. This reliance reintroduced a security risk that originally prompted the use of threshold encryption in the first place. To this end, fully-fledged threshold schemes have been proposed, (e.g., [DK01]). In these schemes, the involvement of a trusted dealer is entirely eliminated, and the generation and distribution of keys are carried out by the participating parties themselves.

Similar to other RSA-based primitives, like signature [RSA78] and verifiable delay function (VDF) [BBBF18] schemes, the public key of the Paillier encryption scheme consists of a modulus  $N$  that is the product of two large prime numbers  $P$  and  $Q$ . Therefore, many works (see [BDF+23] and references within)

deal with distributed RSA modulus as a stand alone and independent building block, leaving additional necessary cryptographic material to be generated by the specific application, be it the RSA signature, RSA encryption, VDF, or Paillier encryption schemes. The additional cryptographic material may be different from scheme to scheme. In the context of threshold encryption, without employing some verifiability mechanism it might not be possible to tell whether a decryption share was computed correctly or not. To this end, proposals for threshold Paillier devise such a verifiability mechanism, in the form of a zero knowledge (zk) proof. That is, in addition to the aforementioned decryption share, each party provides a zero knowledge proof for the claim that the decryption share is computed correctly using its secret key share. In that sense, the proposed threshold Paillier protocols differ mostly in the way that the zero knowledge proof is implemented, offering a trade off between efficiency and security. Specifically, these protocols offer a trade off in the three metrics below:

- **DKG efficiency** refers to whether Distributed Key Generation is practical.
- **Proof efficiency** refers to the size and the time it takes to generate/verify the zero knowledge proof of the correctness of the decryption share.
- **Strength of assumptions** refers to the cryptographic assumptions underlying the soundness of the proof.

Considering only protocols with a feasible key generation phase, one has to choose between a protocol with an efficient proof system that relies on non-standard assumptions (such as [DJN10] using the assumptions in [DK01]), and a protocol whose proof system is inefficient but relies on standard widely accepted assumptions (e.g., [FS01, HMR+19]). This raises the following question:

*Is it possible for a threshold Paillier encryption scheme to incorporate an efficient key generation and proof while relying solely on standard assumptions?*

In this work we answer this question in the affirmative. We depart from a protocol that has an efficient key generation and proof, but relies on non-standard assumptions, and present a novel reduction technique for the proof of soundness of the zk-proof of correct decryption share. This, for the first time, allows using an efficient version of threshold Paillier without compromising on security.

## 1.1 Previous Work: Efficiency vs Security

In the following, we provide a more detailed overview of the trade-off discussed above, which involves a tension between efficiency and the level of leniency associated with relying on non-standard assumptions. On one extreme, the protocol by Algesheimer et al. [ACS02] allows distributed generation of a *bi-prime* public key  $N = PQ$  consisting of *safe primes* (i.e., with  $(P-1)/2$  and  $(Q-1)/2$  primes). Using safe primes enables an efficient zero knowledge protocol for the correctness of the threshold decryption, while also achieving security under standard assumptions. Generating  $N$  as a product of safe primes is commonly adopted by works that assume a trusted dealer [FPS01], since a dealer can easily generate



such a key. However, *distributed* generation of safe primes remains an infeasible task, which is evident by the fact that [ACS02] has never been implemented.

On the other extreme, [DK01] proposed a protocol for threshold RSA signatures that lowers the bar by generating a public key  $N$  that is a product of ‘general’ primes<sup>1</sup>  $P, Q$  (not necessarily safe), using the same efficient zk-proof as [FPS01]. As mentioned in [DJN10], this also applies to threshold Paillier. However, while avoiding safe primes, the soundness of that zero knowledge proof relies on *non-standard assumptions* (which are discussed in Sect. 1.3).

In order to bridge between the above mentioned extremes, Fouque and Stern [FS01] proposed a new protocol in which the key generation produces primes  $P, Q$  that are only *almost safe primes*, meaning that  $\frac{P-1}{2}$  and  $\frac{Q-1}{2}$  are *B-rough numbers* and *co-prime*. Unlike general primes, the *B-roughness* property facilitates a proof of soundness for the zk-protocol without relying on new assumptions. Nevertheless, while distributed generation of such primes can be done, it is impractical for most use-cases. First, the technique incurs a degradation of the efficiency of the zero knowledge proof. As reported by the authors, the proof efficiency is about  $30\times$  worse compared to [DK01]. In addition, we estimate the distributed key generation phase to be  $1000\times$  slower than [CHI+21].

Another bridging attempt is due to Hazay et al. [HMR+19]. Their protocol, similar to [DK01], builds on a public key  $N$  that is a product of general primes, but uses a different zero knowledge protocol (using the cut-and-choose technique) than the one used in [DK01]. Their proof, while relying on standard assumptions, suffers from poor soundness ( $1/2$ ), which means that it has to be repeated  $\kappa$  times to meet real security requirements.

Since safe prime generation as in [ACS02] is infeasible, we are left with the choice between accepting the extra assumptions in [DK01] and getting an efficient proof (and therefore threshold decryption), or sticking to the widely accepted assumptions but suffering an inefficient proof, as in [FS01] and [HMR+19].

It is worth mentioning a different approach, proposed by Baum et al. [BDTZ16], which removes the zero knowledge proof from threshold Paillier decryption altogether. In order to decrypt the ciphertext  $c = \text{Enc}_{pk}(m; r)$  the parties first reconstruct the randomness  $r$ , which enables extraction of the plaintext  $m$ . However, apart from revealing the randomness  $r$  to the adversary, this approach enables an attacker to anonymously cheat in the reconstruction of  $r$  and deny decryption. Both issues above are problematic in general, yet tolerable in some scenarios<sup>2</sup>.

Lastly we would like to mention a previous work by Seres and Burcsi [SB21] which upgrades the security assumptions of Pietrzak’s proof of exponentiation. While the reductions share some ideas there are meaningful differences. Most

<sup>1</sup> We remark that distributed generation of a product of general primes is due to Boneh and Franklin [BF97] and its improvements [DdSGMRT21, CHI+21, BDF+23].

<sup>2</sup> Specifically, in [BDTZ16] such decryption is happening only in the pre-processing phase of a generic MPC protocol, in which case, neither an abort nor leakage of  $r$  gives the adversary any advantage. Alternatively, the same work proposes a method to avoid denial of decryption at the cost of two additional rounds and assuming the primes are safe, a property we wish to avoid in the first place.

importantly they assume that  $\phi(N)$  does not have factors in a certain range, which is okay in their context since a single party holding the factorization of  $N$  can provide a proof of this fact. In our context it is much more involved to produce such a proof since the factorization of  $N$  is shared among the parties.

## 1.2 Our Contribution

- We present the first practical, efficient, large-scale threshold Paillier encryption protocol, supporting efficient distributed key generation and threshold decryption, built upon the same decisional composite residuosity assumption as the standard Paillier encryption. The protocol is secure in the presence of a malicious adversary who statically corrupts  $t < n$  parties.
- At the heart of our contribution lies a novel proof for the correctness of decryption shares. We make use of the standard proof system of equality of discrete logs (EDL) over the group  $\text{QR}_{N^2}$  of quadratic residues modulo  $N^2$ . When  $N$  is not a product of safe primes, the soundness of the EDL-proof does not suffice to claim correctness of the decryption share. Nevertheless, we show that finding a false decryption share that passes verification reduces to factoring  $N$  (see Theorem 4.5 and the preceding discussion). Such a reduction may be of independent interest: First, the same technique can be used in threshold protocols over RSA groups, such as threshold RSA signature. Second, the requirement of safe primes in various cryptographic primitives can be re-assessed, which is left to future work.<sup>3</sup> The ramification of that reduction is that distributed Paillier key generation can be implemented using *any* distributed bi-prime modulus generation (for ‘general’ primes), and in particular, we can leverage recent advances, like Diogenes [CHI+21], for key generation by thousands of parties.
- In a real-world system required to continuously process many ciphertexts, the parties need to verify decryption shares received from other parties for each of these ciphertexts. Verification of decryption shares from many parties across multiple ciphertexts can easily dominate the overall cost of decryption. To address this issue, we use a batching technique due to [APB+04], which allows a prover to *batch* several decryption shares into a single proof. Specifically, proving and verifying  $B$  statements using the batched proof system requires a single ‘large’ exponentiation and  $\mathcal{O}(B)$  ‘small’ exponentiations, rather than  $\mathcal{O}(B)$  large exponentiations (where ‘large’ refers to the size of the shares, e.g., 4096 bits, and ‘small’ refers to the computational security parameter, e.g., 128 bits). The original proof of this batching technique has also required safe primes. We are able to apply our techniques to alleviate this requirement. This results in the first batched proof system for threshold Paillier decryption when  $N$  is not a product of safe primes. Overall it enables a significantly more efficient proof system (as demonstrated in Sect. 5). It reduces the total time

---

<sup>3</sup> That being said, while similar techniques may be applied to remove the requirement of safe primes in other cases as well, in some protocols the requirement that the primes are safe might be crucial, so every protocol must be analyzed on its own.

complexity of threshold decryption to roughly that of a semi-honest model with no proof system. E.g., for 1000 parties proof overhead is about 5%.

Although our contributions are concentrated around threshold decryption, we briefly discuss the protocol for distributed key generation as well in Sect. 3.1, for completeness of the exposition.

*Performance comparison.* In Table 1 we compare the performance between this work and previous works that rely on standard assumptions only, namely [FS01] and [HMR+19]. As the performance is dominated by exponentiations by a large exponent (thousands bit-length) the table focuses on that metric. Note that the large exponents in all the protocols are of the same size. Importantly, our protocol is the only one that supports batch decryption (which essentially requires batching of proofs). Thus, in [FS01] and [HMR+19] the overhead of decrypting a batch of ciphertexts grows linearly with the batch size, whereas in our protocol the overhead remains constant.

**Table 1.** The number of exponentiations required for proof generation and verification in each protocol, where  $\kappa$  represents computational security (i.e.,  $2^\kappa$  is infeasible), and  $\sigma$  represents statistical security (i.e.,  $2^{-\sigma}$  is negligible). For [FS01], we utilize the guaranteed  $B$ -roughness of  $\frac{P-1}{2} \cdot \frac{Q-1}{2}$ .

Work	Number of Exponentiations		Supports batching
	In general	When $\kappa = 128, \sigma = 40, B = 2^{16}$	
[FS01]	$\approx \frac{2\kappa\sigma}{\log^2 B}$	64	No
[HMR+19]	$\kappa$	128	No
<b>This work</b>	<b>2</b>	<b>2</b>	Yes

### 1.3 Technical Overview

Let us begin with the high-level overview of the threshold Paillier decryption. As mentioned above, Paillier’s public key is a modulus  $N$  that is a product of two large primes  $P$  and  $Q$ . The secret key  $d$  is derived from these primes, and it is assumed to be computationally infeasible to obtain it from  $N$ . In threshold Paillier key generation protocols, the parties first obtain a sharing of  $P$  and  $Q$ , and then derive a sharing of  $d$  as well as the product  $N = PQ$  in plain. To be more specific, the generated primes  $P$  and  $Q$  are required to satisfy  $\gcd(\phi(N), N) = 1$ , where  $\phi(N) = (P - 1)(Q - 1)$ . Using this fact, the secret key is computed<sup>4</sup> by  $d = \phi(N) \cdot [\phi(N)^{-1} \bmod N] \in \mathbb{Z}$  and shared among the parties using a secret sharing scheme over the integers<sup>5</sup>, such that party  $P_j$  obtains a share  $d_j$ . In addition to the public modulus  $N$ , the parties generate a public verification key

<sup>4</sup> There are several choices for the exact form of the secret key, which are all variants of the one described above.

<sup>5</sup> Some works assume secret sharing over the ring  $\mathbb{Z}_{N\phi(N)}$  but this is harder to achieve without a trusted dealer.

$v_j$  for every  $P_j$ , such that  $v_j = g^{d_j}$  for some basis element  $g$  from the group of quadratic residues modulo  $N^2$  (denoted  $\text{QR}_{N^2}$ ). Then, when the parties agree to decrypt a ciphertext  $\text{ct}$ , party  $P_j$  sends the decryption share  $\text{ct}_j = \text{ct}^{d_j}$  along with a zero-knowledge proof that  $\text{ct}_j$  is computed correctly using the public  $\text{ct}$  and the secret  $d_j$ .<sup>6</sup> This proof is a proof of *equality of discrete logs* of the values  $\text{ct}_j$  and  $v_j$  over  $\text{QR}_{N^2}$ , with respect to the bases  $\text{ct}$  and  $g$ . That is, if  $P_j$  computes its decryption share correctly then  $\log_{\text{ct}}(\text{ct}_j) = \log_g(v_j) = d_j$ .

In the following we present in more detail why safe primes are powerful for efficient proofs, and later we explain how we achieve the same efficiency without using safe primes and without resorting to additional assumptions.

Suppose that the generated modulus  $N$  is a product of two safe primes  $P, Q$ , meaning that  $P' = \frac{P-1}{2}$  and  $Q' = \frac{Q-1}{2}$  are primes as well. There are three main benefits from the assumption that  $P'$  and  $Q'$  are primes:

1. In the context of proofs for equality of discrete logs, it is commonly assumed that the group is cyclic. For any pair  $P, Q$  of distinct safe primes, the group  $\text{QR}_{N^2}$  is guaranteed to be cyclic, since  $\text{QR}_{N^2} \cong \text{QR}_{P^2} \times \text{QR}_{Q^2}$  and the orders of  $\text{QR}_{P^2}$  and  $\text{QR}_{Q^2}$  are co-prime. However, in the general case these orders may share a common factor and the group  $\text{QR}_{N^2}$  may not be cyclic.
2. When the group  $\text{QR}_{N^2}$  happens to be cyclic, meaning that there exist an element  $g$ , called a generator, such that every element in  $\text{QR}_{N^2}$  equals  $g^x$  for some  $x$ . In fact, the probability of a random element  $g \leftarrow \text{QR}_{N^2}$  to not be a generator, is close to the probability of guessing one of the factors of  $N$ . Having a generator helps when arguing security for zero knowledge protocols. However, in the general case (where the primes are not safe), even when  $\text{QR}_{N^2}$  is cyclic, the probability of a random element to be a generator is not negligible. This problem is exacerbated by the fact that no known algorithm exist for finding a generator or determine if an element is one.
3. The standard proof of soundness (see [FPS01]) obtains an equation of the form  $x^e = 1 \pmod{N^2}$  for a small  $e$  ( $\ll \min\{P', Q'\}$ ), and concludes that  $x = 1$  since  $e$  is necessarily co-prime with  $\phi(N^2)/4 = NP'Q'$ . This conclusion cannot be made when  $P'$  and  $Q'$  are allowed to be composite numbers.

These three benefits of safe primes are also drawbacks of general primes. The work [DK01] overcomes these drawbacks and applies the same zero knowledge proof as in [FPS01], but does not assume that the primes are safe and therefore relies on the following non-standard assumptions:

1. It is computationally hard to compute an element  $a \in \mathbb{Z}_N^*$  such that  $a \neq \pm 1 \pmod{N}$  and the order of  $a$  is not divisible by the largest factor of  $\phi(N)$ .
2. Random elements in  $\text{QR}_N$  are indistinguishable from those of maximal order.

While these assumptions have not been proven insecure to date, relying on non-standard cryptographic assumptions that have received very little attention can potentially pose a risk in practice.

<sup>6</sup> When the threshold is smaller than the number of parties each exponent is multiplied by the appropriate Lagrange coefficient.

In [FS01] mentioned earlier, which aims at avoiding extra assumptions without requiring the primes to be safe, the efficiency is degraded for two reasons. First, the protocol has to be repeated with many bases  $g_1, g_2, \dots$ , which *together* generate  $\text{QR}_{N^2}$  with overwhelming probability. Second, the protocol requires  $\frac{P-1}{2}$  and  $\frac{Q-1}{2}$  to be  $B$ -rough (i.e., to have all prime factors larger than  $B$ ), and the soundness depends on  $B$ . Since all known practical key generation protocols result in quite a small  $B$ , soundness must be amplified via parallel repetitions.

**Our Approach.** In this work we take the same approach as in [DK01] (as it applies to the threshold Paillier cryptosystem presented in [DJN10] with  $s = 1$ ), but *remove their extra assumptions*. Specifically, we show that the efficient zk-proof of equality of discrete logs over  $\text{QR}_{N^2}$  first used in [FPS01] for a modulus  $N$  that is product of safe-primes, suffices even when  $N$  is a product of general primes.<sup>7</sup> By setting minimal and practically achievable properties to the primes, we overcome the above three drawbacks by using the following observations.

1. Even though  $P, Q$  are not safe-primes, with high (but not overwhelming) probability  $\frac{P-1}{2}$  and  $\frac{Q-1}{2}$  are co-prime, and therefore  $\text{QR}_{N^2}$  is a cyclic group of order  $\frac{N\phi(N)}{4}$ . During the key generation protocol the parties will reject prime candidates that do not meet the requirement that  $\frac{P-1}{2}$  and  $\frac{Q-1}{2}$  are co-prime. The rejection of prime candidates that do not satisfy that condition incurs only a small constant factor overhead (about  $1.5\times$  as calculated in Appendix E.6) to the key generation protocol.
2. Even when  $P, Q$  are not safe, the order of a random element  $g \in \text{QR}_{N^2}$  is ‘close enough’ to the order of  $\text{QR}_{N^2}$ , and so for our purpose it can be used as if it was a generator. Specifically, we prove that with overwhelming probability, the order of a randomly sampled  $g \in \text{QR}_{N^2}$  is at least  $|\text{QR}_{N^2}|$  divided by a sufficiently smooth number (whose prime factors are all small).
3. For similar reasons, instead of obtaining the equation  $x^e = 1 \pmod{N^2}$  we obtain  $x^e \cdot \eta = 1 \pmod{N^2}$ , where  $\eta \in \text{QR}_{N^2}$  has a smooth order  $\delta$ . This means that  $x^{e \cdot \delta} = 1 \pmod{N^2}$ . Then, we divide the proof into two cases: If  $x^e = 1$  (as in the case where  $g$  is a generator), then we can find the factorization of  $e$  since  $e$  is small, from which we can find the factorization of  $N$  using classical techniques, as long as  $x \neq 1$ . Otherwise,  $x^e \neq 1 \pmod{N^2}$  is an element of small smooth order and so we can employ Pollard’s  $p - 1$  method in order to factor  $N$ .

Notice that the above ideas would result in an efficient reduction, albeit non-polynomial, and thus would only break the sub-exponential factoring problem. Our simulation carefully chooses challenges such that the size of  $e$  has a bound

---

<sup>7</sup> We do require  $N = PQ$  to satisfy  $\gcd(P - 1, Q - 1) = 2$ , which is a much weaker condition than the common requirement that  $(P - 1)/2$  and  $(Q - 1)/2$  are prime. This property has a small impact on the efficiency of the key generation protocol, does not affect the efficiency of the threshold decryption, and does not introduce additional cryptographic assumptions.

dependent on the chance of success of the PPT adversary rather directly on the security parameter, resolving this issue. Also, the smoothness of the order of  $x^e$  depends on the statistical security parameter. We reduce this to be a constant by using multiple ( $c$ ) bases  $g_1, \dots, g_c$ . As a result, with overwhelming probability with respect to the statistical security parameter  $\sigma$ , there exist at least one base  $g_i$  with order  $|\text{QR}_{N^2}|$  divided by a  $\sigma_0$ -smooth number. We set  $\sigma_0$  to be a constant, and increase  $c$  to increase statistical security. The reduction is analyzed such that one may estimate concrete statistical and computational security based on different parameters of the scheme (see Table 2). Carefully analysing the reduction, we see that setting  $\sigma_0 = 40$  yields an efficient reduction when setting concrete parameters, as  $2^{40}$  exponentiations is feasible to date. This means that if an efficient cheating prover exists, in concrete parameters, our reduction suggests a practical factoring algorithm.

As we are also concerned with concrete security, we provide two reductions, one to the uniform factoring problem and another for the non-uniform factoring problem. The reason is that although we do not require non-uniformity to prove that our scheme is secure asymptotically, when fixing concrete parameters, the reduction against a non-uniform PPT adversary is more tight. Specifically, an adversary  $\mathcal{P}^*$  that breaks our scheme in time  $T(\kappa)$  with probability  $\varepsilon(\kappa)$  is reduced to a non-uniform adversary  $\mathcal{A}$  that breaks the factoring game above in expected time  $T_{\text{NU}}(\kappa)$  with probability  $\varepsilon_{\text{NU}}(\kappa)$ , such that  $\frac{T_{\text{NU}}(\kappa)}{\varepsilon_{\text{NU}}(\kappa)} = \frac{T(\kappa)}{\varepsilon(\kappa)} \cdot \text{poly}(\kappa)$ . That is, the reduction is linear in  $T/\varepsilon$ . On the other hand, the reduction against uniform PPTs takes  $\frac{T_{\text{U}}(\kappa)}{\varepsilon_{\text{U}}(\kappa)} = \frac{T(\kappa)}{\varepsilon^2(\kappa)} \cdot \text{poly}(\kappa)$ , which is the complexity of computing the advice tape for the non-uniform reduction. While this reduction is still polynomial, when fixing concrete parameters, the computational security parameter has to be doubled. Due to the sub-exponential hardness of factoring, this requires increasing the modulus size by  $8\times$ , and so exponentiation will cost at-least  $64\times$  more. It would also significantly impact DKG performance.

Another subtle issue arises when applying the Fiat-Shamir transform to the batched proof, which is a 5-round protocol. In general this may significantly degrade soundness. We show this is not the case for the batched protocol by applying concepts from [AFK22] in Appendix I.

## 1.4 Organization

In Sect. 2 we present our notation and the mathematical background that is necessary in order to understand the rest of the paper. A brief reminder of basic number theoretic facts can be found in Appendix A. In Appendix B we recall Shamir Secret Sharing (SSS) over a field along with proving improved bounds on the size of SSS shares over the integers. Background on the Paillier encryption scheme can be found in Appendix C. In Sect. 3 we present secure threshold Paillier. Further details are in Appendix E, including experiments to estimate the overhead of our adjustment to the key generation protocol, and Appendix F describes our UC simulation. In Sect. 4 we present the zero knowledge protocol for equality of discrete logs over  $\text{QR}_{N^2}$ , and provide our new proof of security.

Omitted proofs are in Appendix G. Subsects. 4.4 and 4.6 present optimization techniques, and in Appendix H we further discuss batch verification optimization and multi-exponentiation. Finally, we report our experiments in Sect. 5.

## 2 Preliminaries

*General Notation.* We let  $\mathbb{N}, \mathbb{Z}, \mathbb{Z}_m$  denote the set of natural numbers excluding 0, integers, and integers modulo  $m$ , respectively. In addition, we denote by  $\mathbb{Z}_m^*$  the multiplicative group modulo  $m$ . We denote by  $\text{primes}$  and  $\text{primes}_m$  the set of all prime numbers and the set of prime numbers smaller than  $m$ , respectively. For  $a, b \in \mathbb{Z}$  we denote by  $[a], [a, b], [a, b]$  and  $(a, b)$  the sets  $\{1, \dots, a\}, \{a, \dots, b\}, \{a, \dots, b-1\}$  and  $\{a+1, \dots, b-1\}$ , respectively. We denote by  $X \leftarrow \Omega$  a uniform sampling from a set  $\Omega$ . We use  $\kappa$  and  $\sigma$  to denote computational and statistical security parameters, respectively. We denote by  $\sigma_0$  a preliminary statistical security parameter which is upgraded to  $\sigma := c \cdot \sigma_0$ . We denote by  $\text{time}(A(x_1, x_2, \dots))$  the run time of an algorithm  $A$  on inputs  $(x_1, x_2, \dots)$ . We may denote a vector of group elements by  $(g_1, \dots, g_c) := \mathbf{g} \in \mathcal{G}^c$  where  $\mathbf{g}^r := (g_1^r, \dots, g_c^r)$ .

*Mathematical Background.* We refer to Appendix A for preliminary mathematical background. Below we list three useful definitions.

**Definition 2.1 ( $\beta$ -Smooth Number).** For  $\beta \in \mathbb{N}$ , an integer  $k$  is called  $\beta$ -smooth if all the prime factors of  $k$  are smaller than  $\beta$ .

**Definition 2.2 (Safe Prime).** A prime number  $p$  is called safe if  $\frac{p-1}{2}$  is prime.

**Definition 2.3 (Conforming Bi-Prime).** An integer  $N$  is a conforming bi-prime if there exist two primes  $P, Q$  such that  $N = PQ$ ,  $P = Q = 3 \pmod{4}$ ,  $\gcd(N, \phi(N)) = 1$ , and  $\gcd(P-1, Q-1) = 2$ .

*Groups.* We use multiplicative notation for groups. Let  $\mathcal{G}$  be a finite abelian group. For  $g_1, \dots, g_k \in \mathcal{G}$  we denote by  $\langle g_1, \dots, g_k \rangle$  the subgroup  $\mathcal{H} \subseteq \mathcal{G}$  generated by  $g_1, \dots, g_k$ . That is,  $\mathcal{H} = \{\prod_{i=1}^k g_i^{\alpha_i}\}_{\alpha_i \in \mathbb{Z}, i \in [k]}$ . We denote by  $|\mathcal{G}|$  and  $\text{ord}(g)$  (or  $|\langle g \rangle|$ ) the order of (number of elements in)  $\mathcal{G}$  and  $\langle g \rangle$ , respectively. An element  $y \in \mathcal{G}$  is a *quadratic residue* if there exists an  $x \in \mathcal{G}$  with  $x^2 = y$ . For abelian groups, the set of quadratic residues forms a subgroup. For an integer  $m > 2$ , we denote by  $\text{QR}_m$  the subgroup of quadratic residues in  $\mathbb{Z}_m^*$ . By Lagrange's theorem, if  $\mathcal{G}$  is a group and  $\mathcal{H} \subseteq \mathcal{G}$  then  $|\mathcal{H}|$  divides  $|\mathcal{G}|$ .

**Definition 2.4 (Statistical Distance).** Let  $X, Y : \Omega \rightarrow [M]$  be two random variables. The statistical distance between  $X, Y$ , denoted  $\text{SD}(X, Y)$ , is defined as  $\text{SD}(X, Y) := \frac{1}{2} \sum_{w \in \Omega} |\Pr[X = w] - \Pr[Y = w]|$ . If  $\text{SD}(X, Y) = \text{neg}(\kappa)$  we say that the distributions are statistically indistinguishable and denote  $X \stackrel{s}{\equiv} Y$ .

*Hardness Assumptions.* Let  $\text{GenModulus}$  be a polynomial time algorithm that, on input  $1^\kappa$ , outputs  $(N, P, Q)$  where  $N = PQ$ , and  $N$  is a conforming bi-prime except with probability negligible in  $\kappa$ .

**Definition 2.5 (DCR).** We say that the decisional composite residuosity (DCR) problem is hard relative to  $\text{GenModulus}$  if for all probabilistic polynomial time algorithms  $\mathcal{A}$  there exists a negligible function  $\text{neg}$  such that

$$|\Pr[\mathcal{A}(N, [r^N \bmod N^2]) = 1] - \Pr[\mathcal{A}(N, r) = 1]| \leq \text{neg}(\kappa)$$

where the probabilities are taken over  $N \leftarrow \text{GenModulus}(1^\kappa)$  and  $r \leftarrow \mathbb{Z}_{N^2}^*$ .

**Definition 2.6 (Factoring).** We say that the factoring problem is hard relative to  $\text{GenModulus}$  if for all (uniform / non-uniform) probabilistic polynomial time algorithms  $\mathcal{A}$  there exists a negligible function  $\text{neg}$  such that

$$\Pr_{N \leftarrow \text{GenModulus}(1^\kappa), (P', Q') \leftarrow \mathcal{A}(N)} [P' \cdot Q' = N \wedge P', Q' \notin \{1, N\}] \leq \text{neg}(\kappa).$$

*Remark 2.1.* In the classic factoring problem  $\text{GenModulus}$  is defined by choosing  $P, Q$  as independently, uniformly random  $\ell$ -bits random primes. In the case of distributed key generation we get a different  $\text{GenModulus}$ . Notably, [BF97] provide a reduction between the two cases.

## 2.1 Shamir Secret Sharing over the Integers

Shamir threshold secret sharing over a field [Sha79] is presented in Appendix B.1. We present its extension over the integers [NS10, Rab98, VAS19] below.

Let  $s \in \mathbb{Z} \cap [-b, +b]$  be a secret. Define  $\Delta_n = n!$  and define some bound  $I(\sigma, n, b)$  on the (absolute value of the) coefficients of the polynomial. Until recently (see, e.g., [VAS19]), the bound  $I(\sigma, n, b) = 2^\sigma \cdot \Delta_n^2 \cdot b$  was used. However, following [BDO23], we provide a tighter bound for  $I(\sigma, n, b)$  in Appendix B.2.

The algorithm  $\text{Share}_{t,n}(s)$  picks  $a_1, \dots, a_t \leftarrow [-I(\sigma, n, b), +I(\sigma, n, b)]$  and outputs  $([s]_1, \dots, [s]_n)$  where  $[s]_j = p(j)$  and  $p(x) = \Delta_n \cdot s + \sum_{i=1}^t a_i \cdot x^i$ .

Reconstruction works as follows. Given a set  $T \subset [n]$  of  $t+1$  distinct elements and given points  $\{(j, [s]_j)\}_{j \in T}$ , we have  $p(0) = \sum_{j \in T} \lambda_{T,j}^0 \cdot [s]_j = \Delta_n s$ , where  $\lambda_{T,j}^v$  is the Lagrange coefficient corresponding to point  $j \in T$ , to restore the polynomial  $p$  evaluation at  $v$  (see Appendix B). However, since the Lagrange coefficients might not be in  $\mathbb{Z}$ , we multiply them first by  $\Delta_n$ , and get

$$\sum_{j \in T} \Delta_n \lambda_{T,j}^0 \cdot [s]_j = \Delta_n^2 \cdot s. \quad (1)$$

For an  $(n, t)$ -threshold Shamir sharing over the integers of secret  $s \in [-b, +b]$ , we denote the upper bound on the absolute value of the shares on  $s$  by  $D(\sigma, n, t, b)$ , which is:  $D(\sigma, n, t, b) = \Delta_n \cdot b + \sum_{i=1}^t I(\sigma, n, b) \cdot n^i \leq \Delta_n \cdot b + 2n^t I(\sigma, n, b)$ .



## 2.2 Zero Knowledge

A *zero knowledge protocol* between a prover  $\mathcal{P}$  and a Verifier  $\mathcal{V}$  is a protocol in which the prover convinces the verifier of some fact regarding a public instance without revealing any information to the verifier beyond the correctness of this fact. A formulation of such property is typically done via three definitions, namely *completeness*, *soundness* and *zero-knowledge*. Completeness means that if both the prover and the verifier act according to the protocol then the verifier will accept. Soundness means that it's computationally infeasible for a cheating prover to convince an honest verifier of a false fact. Lastly zero-knowledge means that the verifier does not learn any new information about the instance except the fact. In our context, we are specifically interested in *Special honest verifier zero-knowledge (SHVZK)* defined below.

Let  $R \subset \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^*$  be a ternary relation. Given a public parameter  $\tau$ , we call  $w$  a witness for instance  $x$  if  $(\tau, x, w) \in R$ . We may fix  $\tau$  and write  $(x, w) \in R[\tau]$  for the public parameter-dependent relation. Define  $L_R[\tau]$  to be the set of inputs  $x$  for which there exists a witness  $w$  such that  $(x, w) \in R[\tau]$ .

**Definition 2.7.** A protocol  $\pi = (\text{Setup}, \mathcal{P}, \mathcal{V})$  with a PPT setup between a prover  $\mathcal{P}$  and a verifier  $\mathcal{V}$  is a zero-knowledge argument of relation  $R$  if it satisfies:

- **Completeness.** For every  $\tau \leftarrow \text{Setup}(1^\kappa, 1^\sigma)$ , if  $\mathcal{P}$  and  $\mathcal{V}$  follow protocol  $\pi$  on input  $\tau, x$  and private input  $w$  to  $\mathcal{P}$ , where  $(x, w) \in R[\tau]$ , then  $\mathcal{V}$  accepts.
- **Soundness.** For any stateful polynomial time prover  $\mathcal{P}^* = (\mathcal{P}_1^*, \mathcal{P}_2^*)$  (i.e.,  $\mathcal{P}_1^*$  and  $\mathcal{P}_2^*$  have an access to the same state)

$$\Pr_{\substack{\tau \leftarrow \text{Setup}(1^\kappa, 1^\sigma) \\ x \leftarrow \mathcal{P}_1^*(1^\kappa, 1^\sigma, \tau)}} [\forall w : (x, w) \notin R[\tau] \wedge (\mathcal{P}_2^*(1^\kappa, 1^\sigma, \tau) \leftrightarrow \mathcal{V}(\tau, x)) = 1] \leq \text{neg}(\kappa).$$

While the standard definition of soundness requires the above to hold for every  $x$ , here we require that it holds only for instances  $x$  that are output by the dishonest prover  $\mathcal{P}^*$  itself, hinting to the difficulty of finding an instance  $x$  for which the equation does not hold.

- **Special honest verifier zero-knowledge.** There exists a PPT simulator  $\mathcal{S}$  such that for every  $z \in \{0, 1\}^*$  the view of the verifier  $\text{View}_{\mathcal{V}}[\mathcal{P}(\tau, x; w) \leftrightarrow \mathcal{V}(\tau, x; z)]$  is statistically indistinguishable from  $\mathcal{S}(\tau, x, z)$ .

It is common to turn a zero-knowledge protocol into a non-interactive zero-knowledge protocol using the Fiat-Shamir (FS) transform [FS87, CCH+18, AFK22], for which it is sufficient to consider only an honest verifier. We refer the reader to [GO94] for further discussion on ZK protocols definitions.

## 3 Our Construction

We refer the reader to Appendix C for the definition of the Paillier additively homomorphic encryption scheme. In this section we present a construction for

a threshold Paillier cryptosystem, and its comprehensive description appears in Appendix E. As shown in Appendix F.2, the protocol realizes the ideal threshold Paillier Functionality F.1 ([HMR+19]).

### 3.1 Key Generation

The key generation part of the protocol is a composition of 2 steps: (i) an adaptation of Diogenes [CHI+21] RSA key-generation for generating  $N$ ; (ii) an adaptation of [HMR+19] to generate the verification keys and key-shares that are used in Paillier threshold decryption protocol. [HMR+19] first generates a non-standard verification key, namely an El-Gamal encryption of the key share. Nevertheless, they refer to a zk proof that ties this verification key with the traditional one that we use. A direct approach avoiding the El-Gamal encryption could be considered, but we chose to use the protocol as-is, utilizing the universal composability guarantees. Notably, modulus generation is the bottleneck regardless. Importantly, this zk proof does not rely on  $N$  being composed of safe-primes (and in turn is somewhat costly, but far less than generating  $N$ ). Hence, proving validity of decryption shares with respect to the traditional verification keys remains the core issue addressed in this work, in Sect. 4.

We henceforth give a short summary of the ideas used for key generation which are presented in further detail in Appendix E. Many protocols for distributed generation of bi-prime (or RSA) modulus were proposed (e.g. [BF97, DdSGMRT21, BDF+23]). In Diogenes the parties obtain a bi-prime  $N = PQ$  for some large primes  $P$  and  $Q$ , but in our context (threshold Paillier) we need  $N$  to be a *conforming* bi-prime (Definition 2.3). Namely,  $N = PQ$  should satisfy: (1)  $P = Q = 3 \pmod{4}$ , (2)  $\gcd(N, \phi(N)) = 1$ , and (3)  $\gcd(P - 1, Q - 1) = 2$ . Generating a conforming bi-prime requires only a minor modification to Diogenes that will not affect security. Conditions (1) and (2) are already satisfied by Diogenes. We ensure that condition (3) is satisfied by invoking the GCD test sub-protocol, which receives one additively shared secret and one public value as input and outputs their GCD. Since both  $P - 1$  and  $Q - 1$  are secrets, we need to manipulate the inputs in order to verify the third condition. Applying the same trick as in [FS01], we note that  $\gcd(P - 1, Q - 1) = \gcd(P - 1 + (Q - 1)P, Q - 1) = \gcd(N - 1, Q - 1)$ . Therefore, we run the GCD test on the secret value  $Q - 1$  and the public value  $N - 1$ . Notably, this additional check is efficient<sup>8</sup> and it fails with probability  $\approx 0.33$ , as predicted and tested in Appendix E.6. Upon failure the protocol is restarted.

Then, using the obtained values above, the parties generate the secret key  $d$  and the verification keys. Recall that the secret key should satisfy  $d = 0 \pmod{\phi(N)}$  and  $d = 1 \pmod{N}$ . As  $\phi(N)$  is already shared by the parties, such secret key can be obtained by computing  $d = \phi(N)[\phi(N)^{-1} \pmod{N}] \in \mathbb{Z}$ , which satisfies both constraints:  $d = 0 \pmod{\phi(N)}$  as it is a multiple of  $\phi(N)$ , and  $d = 1$

<sup>8</sup> Since Diogenes protocol applies thousands of GCD tests internally, the computational overhead of  $\gcd(N - 1, Q - 1)$  is negligible ( $< .1\%$ ).

mod  $N$  as  $\phi(N) \in \mathbb{Z}_N^*$  (guaranteed by the fact that  $N$  is a conforming bi-prime) and so  $\phi(N) \cdot \phi(N)^{-1} = 1 \pmod N$ .

The parties obtain a sharing of  $d$  using standard techniques, see Hazay et al. [HMR+19, Appendix C.2]. In the same work [HMR+19] it is shown how the parties obtain the verification keys  $v_j$  as well; below we briefly describe how it works. During the key generation phase the parties obtain  $c_j = \text{Enc}_{pk}(d_j)$  for every  $j$ , where  $\text{Enc}$  is the El-Gamal encryption scheme and  $pk$  is a joint encryption public key that was previously generated by the parties. Then, the parties sample a vector of random bases  $\mathbf{g} = (g_1, \dots, g_c) \in \text{QR}_{N^2}^c$ . Then, each party publishes a corresponding vector of verification keys  $\mathbf{v}_j = \mathbf{g}^{d_j}$ , and proves that they correspond to the plaintext  $d_j$  encrypted under  $c_j$ . The language  $L_{\text{EQ}}[N] = \{(c, v)\}$ , used for binding El-Gamal encryptions to verification keys, is formally described in [HMR+19, Section 3]. The corresponding zero-knowledge protocol  $\Pi_{\text{EQ}}$  is presented in [CKY09]. Note that, we could have used the exact same technique of [HMR+19] for our threshold Paillier protocol, namely, whenever a party sends a decryption share  $\text{ct}_j = \text{ct}^{d_j}$ , it also provides a proof that  $c_j$  is an encryption of  $\log_{\text{ct}}(\text{ct}_j)$ . This, however, would result with an inefficient threshold decryption protocol, as such proof is at least  $\times 64$  more expensive than the proof of the language  $L_{\text{EDL}^2}$  that we use (see Sect. 4).

### 3.2 Threshold Decryption

Here we present a standard approach for secure threshold decryption of Paillier ciphertexts by [DJN10]. Recall (see Sect. 3) that the parties securely generated a conforming bi-prime  $N$ , hold a Shamir sharing over the integers  $[d] = \{d_j\}_{j \in T}$  of the secret decryption key  $d$  satisfying  $d \equiv 0 \pmod{\phi(N)}$  and  $d \equiv 1 \pmod N$ . In addition, a list of random elements  $\mathbf{g} \in \text{QR}_{N^2}^c$  is generated, and the verification keys  $\mathbf{v}_j = \mathbf{g}^{d_j}$  for each party is also published.

In the following we assume that the parties behave honestly and later we describe how to handle malicious behaviour. Given a ciphertext  $\text{ct} = \text{Enc}(\text{pt}; r) \in \mathbb{Z}_{N^2}^*$ , party  $P_j$  broadcasts its decryption share  $\text{ct}_j = \text{ct}^{2\Delta_n d_j}$  (recall that  $\Delta_n = n!$ ). Given  $\text{ct}_j$  for all  $j \in T$ , where  $T \subset [n]$  and  $|T| = t+1$ , decrypt by computing:

$$\begin{aligned} \text{ct}' &:= \left[ \prod_{j \in T} \text{ct}_j^{2\Delta_n \lambda_{T,j}^0} \pmod{N^2} \right] = \left[ \prod_{j \in T} (\text{ct}^{2\Delta_n d_j})^{2\Delta_n \lambda_{T,j}^0} \pmod{N^2} \right] & (2) \\ &= \left[ \prod_{j \in T} \text{ct}^{4\Delta_n^2 d_j \lambda_{T,j}^0} \pmod{N^2} \right] = \left[ \text{ct}^{4\Delta_n \sum_{j \in T} \Delta_n d_j \lambda_{T,j}^0} \pmod{N^2} \right] \\ &= \left[ \text{ct}^{4\Delta_n^3 d} \pmod{N^2} \right] = \left[ (\text{ct}^{4\Delta_n^3})^d \pmod{N^2} \right] = \text{Enc} \left( N, 4\Delta_n^3 \text{pt}; r^{4\Delta_n^3} \right)^d, \end{aligned}$$

where the first equality holds since  $\text{ct}_j = \text{ct}^{2\Delta_n d_j}$ , the third follows by Lagrange interpolation in the exponent, since we have  $\left( \sum_{j \in T} \Delta_n d_j \lambda_{T,j}^0 \right) = \Delta_n^2 d$  by Eq. (1), and the last one follows by the encryption definition.

Then, obtain the plaintext by computing

$$\left[ \left( \frac{\text{ct}' - 1}{N} \right) \cdot (4\Delta_n^3)^{-1} \pmod{N} \right] = [\text{pt} \cdot 4\Delta_n^3 \cdot (4\Delta_n^3)^{-1} \pmod{N}] = \text{pt}, \quad (3)$$

as the first equality is derived from the correctness of the Paillier scheme. In the last step (3), we multiply by the multiplicative inverse of  $4\Delta_n^3$  modulo  $N$  to obtain the plaintext  $\text{pt}$ . Correctness therefore holds for any  $\text{pt} \in \mathbb{Z}_N$  and any number of parties  $n < \min\{P, Q\}$ .

*Handling Corrupted Parties.* To detect a malicious party  $P_j$  that sends an incorrect decryption share  $\text{ct}_j$ , we require  $P_j$  to send a zk-proof that  $\text{ct}_j$  is computed correctly using the public base  $\text{ct}$  and the secret share  $d_j$ . As discussed in Sect. 1.1, several approaches were proposed in the literature, and here we follow the one taken by [DJN10]. In more detail, each party  $P_j$  sends along with its decryption share  $\text{ct}_j = \text{ct}^{2\Delta_n d_j}$ , a proof that the discrete log of  $\text{ct}_j$  in the basis  $\text{ct}^2$  equals the discrete log of  $\mathbf{v}_j$  in the basis  $\mathbf{g}$ . In that sense,  $v_j$  is a commitment to  $P_j$ 's secret share  $d_j$ .

We remark that the discrete logarithm equality described above does not assure that  $P_j$  behaves honestly, and a slightly stronger claim needs to be proved. See Sect. 4 for the exact formulation of the language.

## 4 Zero-Knowledge Proof of Equality of Discrete Logs

In this section we present the proof of validity of threshold decryptions by the parties. Notably, defining the appropriate language is somewhat subtle.

### 4.1 Formalizing the Language

We begin by describing the naïve approach, for some intuition:

$$L_{\text{EDL}}^{\text{naive}}[N, \mathbf{g}', \mathbf{a}] = \{(h', b'; x') \mid h', b' \in \mathbb{Z}_{N^2}^* \wedge \mathbf{a} = \mathbf{g}'^{x'} \wedge b' = h'^{x'}\} \quad (4)$$

where  $(h', b')$  is the statement,  $N, \mathbf{g}', \mathbf{a}$  are public parameters, and  $x'$  is a witness. The meaning of these values follows:

1.  $\mathbf{g}'$  are the base elements chosen in the DKG phase.
2.  $\mathbf{a} = \mathbf{v}_j = \mathbf{g}'^{2\Delta_n d_j}$  is the verification key associated with the prover  $P_j$ .
3.  $x' = 2\Delta_n d_j$  is the decryption key-share of  $P_j$ .
4.  $b' = \text{ct}_j = \text{ct}^{2\Delta_n d_j}$  is the claimed partial decryption of the prover  $P_j$ .

The above formalization raises two issues:

1. The Paillier ciphertext  $h' = \text{ct}$  is in  $\mathbb{Z}_{N^2}^*$  and so it would be most natural to prove equality of discrete logs over this group. However, since  $\mathbb{Z}_{N^2}^*$  is not a cyclic group, we work over the subgroup  $\text{QR}_{N^2}$ , which raises another issue: deciding membership to  $\text{QR}_{N^2}$  is assumed to be a computationally hard problem, known as quadratic residuosity problem (QRP).

2. The goal of the zero knowledge proof of the language is to make sure the prover provides  $(h', b')$  such that  $b' = h'^{2\Delta_n d_j}$ . Since  $\mathbf{g}'$  might not contain a generator of  $\text{QR}_{N^2}$ , we may have that for every  $g' \in \mathbf{g}'$   $\text{ord}(g') < |\text{QR}_{N^2}|$ . In such case an adversary may find  $x'' \neq 2\Delta_n d_j \pmod{|\text{QR}_{N^2}|}$  yet  $\mathbf{g}'^{x''} = \mathbf{g}'^{2\Delta_n d_j} = \mathbf{a}$ . This means that for some  $h'$ 's, picking  $b' = h'^{x''}$  gives  $(h', b'; x'')$  in the language although  $b' \neq h'^{2\Delta_n d_j}$ .

To solve the first issue, in the DKG and threshold decryption phases the parties will publish the roots of the elements, and prove statements on their squares (as in [Sho00]). This ensures the whole proof holds over  $\text{QR}_{N^2}$ . We stress that we do not need to roll back and prove any statement in  $\mathbb{Z}_{N^2}^*$ . Instead, after combining the (squared) decryption shares, the reconstructed plaintext will end up being multiplied by 2, and can then be restored by multiplying by  $2^{-1} \pmod N$ . To be more specific, random elements  $\mathbf{g}' \leftarrow (\mathbb{Z}_{N^2}^*)^c$  are sampled and published in the DKG phase, and we set  $\tilde{\mathbf{g}} = \mathbf{g}'^{\Delta_n}$ , and  $\mathbf{g} = \tilde{\mathbf{g}}^2 = \mathbf{g}'^{2\Delta_n} \in \text{QR}_{N^2}^c$ . Similarly, we set  $\tilde{h} = \text{ct}^{2\Delta_n}$  and  $h = \tilde{h}^2 = \text{ct}^{4\Delta_n} \in \text{QR}_{N^2}$ . Finally, we define  $\tilde{b} = \text{ct}_j = \text{ct}^{2\Delta_n d_j} = \tilde{h}^{d_j}$  and  $b = \tilde{b}^2 = h^{d_j}$ . Under the new syntax, we have:

1.  $\mathbf{g}'$  is *some* vector of elements *chosen from*  $(\mathbb{Z}_{N^2}^*)^c$  at DKG, and  $\mathbf{g} = \mathbf{g}'^{2\Delta_n}$ .
2.  $\mathbf{a} = g^{d_j}$  is the verification key  $\mathbf{v}_j = \mathbf{g}'^{2\Delta_n d_j}$  associated with the prover  $P_j$ .
3.  $x = d_j$  is a witness known by the prover  $P_j$ .
4.  $\tilde{h} = \text{ct}^{2\Delta_n}$ .
5.  $\tilde{b} = \text{ct}_j = \text{ct}^{2\Delta_n d_j}$  is the claimed partial decryption of the prover  $P_j$ .

We get that  $\log_h b = \log_{\mathbf{g}} \mathbf{a} = d_j$ , and to make sure that  $\mathbf{g} \in \text{QR}_{N^2}^c, h, b \in \text{QR}_{N^2}$  we define the language with  $\tilde{\mathbf{g}}, \tilde{h}, \tilde{b}$  (instead of  $\mathbf{g}, h, b$ ):

$$L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}] = \{(\tilde{h}, \tilde{b}; x) \mid \tilde{h}, \tilde{b} \in \mathbb{Z}_{N^2}^* \wedge \mathbf{a} = \tilde{\mathbf{g}}^{2x} \wedge \tilde{b}^2 = \tilde{h}^{2x}\}. \quad (5)$$

As a side effect of that formulation we enjoy a shorter witness, that is, in Eq. (4) the witness was  $x' = 2\Delta_n d_j$  whereas here (Eq. (5)) the witness is  $x = d_j$ .

To address the second issue, our soundness argument in Theorem 4.5 implies that  $(\tilde{h}, \tilde{b}; d_j) \in L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}]$  for every  $(\tilde{h}, \tilde{b})$  provided by the (potentially malicious) prover. That is, we essentially prove that the prover must use  $d_j$  as its witness in order for the verifier to accept.

In Sect. 4.2 we describe the setup phase; we present and prove a zero-knowledge protocol for  $L_{\text{EDL}^2}$  in Sect. 4.3; in Sects. 4.4–4.6 we show how to batch the protocol over multiple ciphertexts; and finally we present a non-interactive version of the protocol via the Fiat-Shamir transform in Section 1.

## 4.2 Setup Phase

Let us define the **Setup** algorithm for generating the parameters of the language. We break the setup into two separate phases  $\text{Setup} = (\text{Setup}_1, \text{Setup}_2)$ . The first one is responsible for generating the conforming bi-prime  $N \leftarrow \text{Setup}_1(1^\kappa, 1^\sigma)$ , and is implemented in Protocol E.1. The second one is responsible for generating

all the rest, given  $N$ :  $(\tilde{\mathbf{g}}, \mathbf{a}; x) \leftarrow \text{Setup}_2(1^\kappa, 1^\sigma, N)$ . In the corresponding Protocol E.2, the verification keys  $(\mathbf{v}_j)_j$  for all parties are generated, and each party receives its secret share  $d_j$ . We focus on a single party  $j$  to avoid an extra index  $j$  in the security analysis. Thus,  $\text{Setup}$  takes  $1^\kappa$  and  $1^\sigma$  as inputs and outputs:

- A conforming bi-prime  $N = P \cdot Q$ .
- Random bases  $\tilde{\mathbf{g}}$ , where  $\mathbf{g}'$  is drawn uniformly at random from  $(\mathbb{Z}_{N^2}^*)^c$ , and  $\tilde{\mathbf{g}} = \mathbf{g}'^{\Delta_n}$  and  $\mathbf{g} = \tilde{\mathbf{g}}^2$ . We stress that  $\mathbf{g}$  does not necessarily contain a generator of  $\text{QR}_{N^2}$ .
- The witness  $x$  drawn uniformly from  $[-D, +D]$ . In the scheme, it is a secret share of the private key of a certain party.
- $\mathbf{a} := \mathbf{g}^x$ . In the scheme, it is the verification key.

We note that  $\mathbf{g}'^2$  is a random vector of elements in  $\text{QR}_{N^2}^c$  due to the absence of any known algorithm for computing a generator within this group. This observation underscores the challenge previously mentioned, as conventional protocols for the equality of discrete logarithms with a low soundness error typically assume  $\mathbf{g}$  contains a generator (e.g., [FPS01, DJN10]). We address this challenge by demonstrating that a randomly selected vector of elements contains an “almost generator” of the group with overwhelming probability, as defined below:

**Definition 4.1.** *Let  $\beta \in \mathbb{N}$ , let  $g \in \mathcal{G}$  be an element. We say that  $g$  is a  $\beta$ -almost generator if  $\frac{|\mathcal{G}|}{\text{ord}(g)}$  is a  $\beta$ -smooth number.*

Setting  $\mathbf{g} = (\mathbf{g}'^2)^{\Delta_n}$ , Corollary 4.3 below implies that  $\mathbf{g}$  contains a  $\beta_{\sigma_0}$ -almost generator with probability  $> 1 - 2^{-(\sigma+1)}$  where  $\beta_{\sigma_0} \leq 2^{\sigma_0+3} \log N$ . Apparently, assuming  $\mathbf{g}$  contains an almost generator suffices to obtain a low soundness error.

**Lemma 4.2.** *Let  $\mathcal{G}$  be a cyclic group and let  $g \in \mathcal{G}$  be a uniformly random element, then  $\frac{|\mathcal{G}|}{\text{ord}(g)}$  is  $\beta$ -smooth with probability at least  $1 - \frac{\log \text{ord}(\mathcal{G})}{\beta \log \beta}$ .*

By Lemma 4.2, if we let  $\beta_{\sigma_0} := \lceil 2^{\sigma_0+2} \log(|\mathcal{G}|) \rceil$ , then  $\frac{|\mathcal{G}|}{\text{ord}(g)}$  is  $\beta_{\sigma_0}$  smooth with probability greater than  $1 - 2^{-(\sigma_0+1)}$ .

**Corollary 4.3.** *Let  $\mathcal{G}$  be a cyclic group,  $g \in \mathcal{G}$  be a uniformly random element, and  $\beta > n$ , then  $\frac{|\mathcal{G}|}{\text{ord}(g^{\Delta_n})}$  is  $\beta$ -smooth with probability at least  $1 - \frac{\log \text{ord}(\mathcal{G})}{\beta \log \beta}$ .*

Omitted proofs are in Appendix G. To conclude, the algorithm may fail with probability  $2^{-\sigma}$ . Specifically, failure means that either  $N$  is not a bi-prime or  $\mathbf{g}$  does not contain a  $\beta_{\sigma_0}$ -almost generator. The modulus generation phase can be taken to provide a failure probability of  $2^{-\sigma+1}$  which by the union bound gives the desired failure probability.

### 4.3 The Protocol

The formal description is given in Protocol 4.1. We prove the following.

**Theorem 4.4.** *The protocol  $\Pi_{\text{EDL}^2}$  (Protocol 4.1) is a zero-knowledge argument for relation  $\text{EDL}^2$  (Definition 2.7) under the factoring assumption.*

*Proof.* We show that all properties of a zero-knowledge argument are satisfied.

**PROTOCOL 4.1** (  $\Pi_{\text{EDL}^2}$ : ZKP of Equality of Discrete Logs over  $\text{QR}_{N^2}$  )

**Inputs.**  $\mathcal{P}$  has  $(N, \tilde{\mathbf{g}}, \mathbf{a}, \tilde{h}, \tilde{b}; x)$  and  $\mathcal{V}$  has  $(N, \tilde{\mathbf{g}}, \mathbf{a}, \tilde{h}, \tilde{b})$  where  $(N, \tilde{\mathbf{g}}, \mathbf{a}; x) \leftarrow \text{Setup}(1^\kappa, 1^\sigma)$  and arbitrary  $\tilde{h}, \tilde{b}$ . Denote  $\mathbf{g} = [\tilde{\mathbf{g}}^2 \bmod N^2]$ ,  $h = [\tilde{h}^2 \bmod N^2]$  and  $b = [\tilde{b}^2 \bmod N^2]$ .

**Protocol.**

1.  $\mathcal{P}$  samples  $r \leftarrow [-2^{2\kappa}D, +2^{2\kappa}D)$  and sends  $\mathbf{u} = [\mathbf{g}^r \bmod N^2], v = [h^r \bmod N^2]$  to  $\mathcal{V}$ .
2.  $\mathcal{V}$  samples  $e \leftarrow [0, 2^\kappa)$  and sends  $e$  to  $\mathcal{P}$ .
3.  $\mathcal{P}$  sends  $z = r - e \cdot x \in \mathbb{Z}$  to  $\mathcal{V}$ .
4.  $\mathcal{V}$  verifies that:
  - $h, b, v \in \mathbb{Z}_{N^2}^*$ ,  $\mathbf{u} \in (\mathbb{Z}_{N^2}^*)^c$  (it suffices to check that  $h, b, v, u_1, \dots, u_c \neq 0 \bmod N$  as otherwise we can use these values to factor  $N$ ),
  - $z \in (-D(2^{2\kappa} + 2^\kappa), +D(2^{2\kappa} + 2^\kappa))$ ,
  - $\mathbf{u} = \mathbf{g}^z \cdot \mathbf{a}^e \bmod N^2$  and  $v = h^z \cdot b^e \bmod N^2$ .

*Completeness.* Let  $(N, \tilde{\mathbf{g}}, \mathbf{a}; x)$  be the output of  $\text{Setup}(1^\kappa, 1^\sigma)$ , then, for every  $\tilde{h} \in \mathbb{Z}_{N^2}^*$  and  $\tilde{b} = [\tilde{h}^x \bmod N^2]$  the protocol's transcript is accepting. The range check of  $\tilde{h}, \tilde{b}, \mathbf{u}$  and  $v$  obviously goes through, as well as the range check of  $z$ , which follows immediately from the ranges of  $r, e$  and  $x$ . Then, we have

$$[\mathbf{g}^z \cdot \mathbf{a}^e \bmod N^2] = [\mathbf{g}^{r-ex} \cdot \mathbf{g}^{ex} \bmod N^2] = [\mathbf{g}^r \bmod N^2] = \mathbf{u}, \text{ and}$$

$$[h^z \cdot b^e \bmod N^2] = [h^{(r-ex)} \cdot h^{ex} \bmod N^2] = [h^r \bmod N^2] = v.$$

*Special Honest-Verifier Zero-Knowledge (SHVZK).* We show that there exists a PPT simulator  $\mathcal{S}$ , such that for every  $(\tilde{h}, \tilde{b}) \in L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}]$  and  $e' \in \{0, 1\}^\kappa$  it holds that  $\mathcal{S}_{(N, \tilde{\mathbf{g}}, \mathbf{a})}(\tilde{h}, \tilde{b}, e') \stackrel{s}{\equiv} \left\{ \text{View}(\mathcal{P}(\tilde{h}, \tilde{b}; x) \leftrightarrow \mathcal{V}(\tilde{h}, \tilde{b}, e')) \right\}$  (see Definition 2.4). The simulator  $\mathcal{S}$  samples  $z' \leftarrow (-D(2^{2\kappa} + 2^\kappa), +D(2^{2\kappa} + 2^\kappa))$ , and computes  $\mathbf{u}' = [\mathbf{g}^{z'} \cdot \mathbf{a}^{e'} \bmod N^2]$  and  $v' = [h^{z'} \cdot b^{e'} \bmod N^2]$ . We argue that the statistical distance between  $(\mathbf{u}, v, e', z)$  (of the real execution) and  $(\mathbf{u}', v', e', z')$  (of the simulation) is negligible in  $\kappa$ . Note that  $(\mathbf{u}, v)$  and  $(\mathbf{u}', v')$  are fully determined by  $(N, \tilde{\mathbf{g}}, \mathbf{a}, \tilde{h}, \tilde{b}, e', z)$  and  $(N, \tilde{\mathbf{g}}, \mathbf{a}, \tilde{h}, \tilde{b}, e', z')$ , respectively. We have that  $z'$  (in the simulation) is uniformly distributed from  $(-D(2^{2\kappa} + 2^\kappa), +D(2^{2\kappa} + 2^\kappa))$  independent of  $e'$ , whereas  $z$  (in the real execution) is computed by  $z = r - e'x$ , where  $x \in [-D, +D]$  and  $r$  is drawn uniformly from  $[-2^{2\kappa}D, +2^{2\kappa}D)$ . Next, we show that  $z$  and  $z'$  are statistically close. The distribution of  $z$  is as follows:

- $x \geq 0$ :  $z$  is uniformly distributed over  $(-D(2^{2\kappa} + 2^\kappa) - e|x|, D(2^{2\kappa} + 2^\kappa) - e|x|)$ .
- $x < 0$ :  $z$  is uniformly distributed over  $(-D(2^{2\kappa} + 2^\kappa) + e|x|, D(2^{2\kappa} + 2^\kappa) + e|x|)$ .

In both cases there are  $2D(2^{2\kappa} + 2^\kappa) - e|x|$  values  $\zeta$  in the range  $(-D(2^{2\kappa} + 2^\kappa), +D(2^{2\kappa} + 2^\kappa))$  for which  $\Pr[z = \zeta] = \Pr[z' = \zeta] = 1 / (2D(2^{2\kappa} + 2^\kappa))$ . For the rest  $e|x| \leq 2^\kappa D$  values  $\zeta$  we have  $\Pr[z = \zeta] = 0$ . Therefore, the distance is  $\frac{1}{2} \sum_{\zeta} |\Pr[z = \zeta] - \Pr[z' = \zeta]| \leq \frac{2^\kappa D}{2D(2^{2\kappa} + 2^\kappa)} < 2^{-\kappa}$ , where  $\zeta$  iterates over  $(-D(2^{2\kappa} + 2^\kappa) - e|x|, +D(2^{2\kappa} + 2^\kappa) + e|x|)$ .

*Soundness.* In our main Theorem 4.5, we prove a stronger property. Namely, we show that no PPT adversary can compute a statement  $(\tilde{h}, \tilde{b})$  that passes verification with non-negligible probability, when  $(\tilde{h}, \tilde{b}; x) \notin L_{\text{EDL}^2}$ . If we assume in contradiction that an adversary breaking the soundness as in Definition 2.7 exists, namely, it can compute a statement  $(\tilde{h}, \tilde{b})$  and pass verification with non-negligible probability, while  $(\tilde{h}, \tilde{b}; x') \notin L_{\text{EDL}^2}$  for any  $x'$ , then it also holds for  $x = x'$  in particular, in contradiction to Theorem 4.5. The stronger property is needed in the UC simulation in Appendix F, since otherwise an adversary could send a wrong decryption share yet pass verification. Notably, there may exist statements  $(\tilde{h}, \tilde{b}; x') \in L_{\text{EDL}^2}$  such that  $(\tilde{h}, \tilde{b}; x) \notin L_{\text{EDL}^2}$  for which an adversary is able to convince the verifier. However, Theorem 4.5 suggests that such statements are computationally hard to find, as it reduces to factoring  $N$ . ■

In what follows, we complete the soundness claim by proving Theorem 4.5, which essentially provides a reduction from factoring to a PPT cheating prover. We break the reduction into two parts. In the first part, Lemma 4.6 provides an algorithm  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  that can find an element  $y \in \text{QR}_{N^2}$  of low order, given a *deterministic* cheating prover. Notably, this does not break the Low Order assumption directly, which requires (a multiple of)  $\text{ord}(y)$  as an additional output. Then, Theorem 4.5 provides a factoring algorithm  $\mathcal{A}$ , given a *probabilistic* cheating prover  $\mathcal{P}^*$ . The factoring algorithm  $\mathcal{A}$  applies several calls to  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$ , sampling the random tape of  $\mathcal{P}^*$  to make it deterministic. Upon success, it extracts  $y \in \text{QR}_{N^2}$  of low order. Using properties of conforming bi-primes, we show that one could use this  $y$  to factor  $N$ .

**Theorem 4.5.** *Let  $\mathcal{P}^* = (\mathcal{P}_1^*, \mathcal{P}_2^*)$  be a stateful probabilistic prover such that*

$$\Pr_{\substack{(N, \tilde{\mathbf{g}}, \mathbf{a}; x) \leftarrow \text{Setup}(1^\kappa, 1^\sigma), \\ (\tilde{h}, \tilde{b}) \leftarrow \mathcal{P}_1^*(x)}} \left[ \begin{array}{l} (\tilde{h}, \tilde{b}; x) \notin L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}] \\ (\mathcal{P}_2^*(x) \leftrightarrow \mathcal{V}(\tilde{h}, \tilde{b})) = 1 \end{array} \left| \begin{array}{l} N \text{ is a conforming} \\ \text{bi-prime and} \\ \tilde{\mathbf{g}}^2 \text{ contains a } \beta_{\sigma_0} \\ \text{almost generator} \end{array} \right. \right] \geq \varepsilon(\kappa),$$

where  $\mathcal{P}_1^*, \mathcal{P}_2^*$  and  $\mathcal{V}$  receive  $(1^\kappa, 1^\sigma, N, \tilde{\mathbf{g}}, \mathbf{a})$  implicitly as inputs, and such that  $\text{time}(\mathcal{P}_1^*) + \text{time}(\mathcal{P}_2^*) \leq T = T(\kappa)$ . Assume further that  $\varepsilon(\kappa) > 2^{-\kappa+3}$ . Then there exists a non-uniform adversary  $\mathcal{A}_{\text{NU}}$  that solves the factorization problem (Definition 2.6) with respect to  $\text{Setup}_1(1^\kappa, 1^\sigma)$ . The adversary  $\mathcal{A}_{\text{NU}}$  first applies  $(\tilde{\mathbf{g}}, \mathbf{a}; x) \leftarrow \text{Setup}_2(1^\kappa, 1^\sigma, N)$ . Then,  $\mathcal{A}_{\text{NU}}$  factors  $N$  with probability  $\geq \varepsilon_{\text{NU}}(\kappa)$  and expected time  $\mathbb{E}[\text{time}(\mathcal{A}_{\text{NU}})] \leq T_{\text{NU}}(\kappa)$  such that  $\frac{T_{\text{NU}}}{\varepsilon_{\text{NU}}} \leq \tilde{\mathcal{O}}(32\kappa c(\frac{T}{\varepsilon} + \beta_{\sigma_0}))$ .

Moreover, there exists a uniform adversary  $\mathcal{A}_{\text{U}}$  that factors with probability  $\geq \varepsilon_{\text{U}}(\kappa)$  and expected time  $\mathbb{E}[\text{time}(\mathcal{A}_{\text{U}})] \leq T_{\text{U}}(k)$  such that  $\frac{T_{\text{U}}}{\varepsilon_{\text{U}}} \leq \tilde{\mathcal{O}}(16\kappa c(\frac{T}{\varepsilon} + \beta_{\sigma_0}))$ .

**Lemma 4.6.** *Let  $\kappa > 0$ , and let  $\varepsilon \geq 2^{2-\kappa}$ . Let  $\mathcal{P}^* = (\mathcal{P}_1^*, \mathcal{P}_2^*)$  be a stateful deterministic prover, and let  $(N, \tilde{\mathbf{g}}, \mathbf{a}; x) \leftarrow \text{Setup}(1^\kappa, 1^\sigma)$  be an output from Setup. Suppose  $N$  is a conforming bi-prime,  $\tilde{\mathbf{g}}^2$  contains a  $\beta_{\sigma_0}$ -almost generator and*

$$\Pr_{(\tilde{h}, \tilde{b}) \leftarrow \mathcal{P}_1^*(x)} \left[ \begin{array}{l} (\tilde{h}, \tilde{b}; x) \notin L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}] \\ (\mathcal{P}_2^*(x) \leftrightarrow \mathcal{V}(\tilde{h}, \tilde{b})) = 1 \end{array} \right] \geq \varepsilon(\kappa),$$



where  $\mathcal{P}_1^*, \mathcal{P}_2^*$  and  $\mathcal{V}$  receive  $(1^\kappa, 1^\sigma, N, \tilde{\mathbf{g}}, \mathbf{a})$  implicitly as inputs. Then there exists an oracle machine  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$ , that is given as input  $(N, \tilde{\mathbf{g}}, \mathbf{a}, \varepsilon)$ , and outputs a non-trivial element  $1 \neq y \in \text{QR}_{N^2}$  and an exponent  $0 < e' < \frac{2}{\varepsilon}$  such that  $\text{ord}(y^{e'})$  is a  $\beta_{\sigma_0}$ -smooth number, with probability at least  $\frac{1}{8c}$ .  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  runs in time  $\leq \tilde{O}(\frac{T}{\varepsilon})$ , where  $T = \text{time}(\mathcal{P}^*(1^\kappa, 1^\sigma, N, \tilde{\mathbf{g}}, \mathbf{a}; x))$ .

We note that Theorem 4.5 and Lemma 4.6 do not assume the cheating prover  $\mathcal{P}^*$  is polynomial-time. Instead, the time complexity and success probability of the factoring reduction  $\mathcal{A}$  are analyzed as a function of  $\mathcal{P}^*$  time complexity  $T$  and success probability  $\varepsilon$ . When applying Theorem 4.5 to a PPT, and  $T, \frac{1}{\varepsilon} \leq \text{poly}(\kappa)$ , we get a PPT reduction. However, one may also be interested in getting a concrete bound on the time to forge a proof, given concrete parameters. To this end, based on NIST's estimations, we can deduce a concrete bound on the complexity of cheating the proof, as done in Table 2.

The first reduction  $\mathcal{A}_{\text{LO}}$  in Lemma 4.6 requires the following Lemma:

**Lemma 4.7.** *Let  $\mathcal{G}$  be cyclic group and let  $|\mathcal{G}| = \prod_{i=1}^k p_i^{r_i}$  where  $p_i \in \text{primes}$  are distinct and  $r_i \in \mathbb{N}$  for all  $i \in [k]$ . Let  $g \in \mathcal{G}$  be an element, and denote  $\text{ord}(g) = \prod_{i=1}^k p_i^{\alpha_i}$  where  $\alpha_i \leq r_i$  for all  $i$ . Then there exists  $\eta \in \mathcal{G}$  such that  $\langle g, \eta \rangle = \mathcal{G}$  and  $\text{ord}(\eta) = \prod_{i:\alpha_i \neq r_i} p_i^{r_i}$ .*

The proof of Lemma 4.7 is in Appendix G. We are now ready to prove the first part of our reduction.

**ALGORITHM 4.2** ( $\mathcal{A}_{\text{LO}}$ : First Part of the Reduction)

**Input.** Algorithm  $\mathcal{A}_{\text{LO}}$  receives as input  $(N, \tilde{\mathbf{g}}, \mathbf{a}, \varepsilon)$  and is given oracle access to a *stateful deterministic* prover  $\mathcal{P}^* = (\mathcal{P}_1^*, \mathcal{P}_2^*)$ .

Algorithm  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  works as follows:

1. **Receiving the claim:**  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  calls  $\mathcal{P}_1^*(1^\kappa, 1^\sigma, N, \tilde{\mathbf{g}}, \mathbf{a}; x)$  and receives  $\tilde{h}$  and  $\tilde{b}$ . Since  $\mathcal{P}_1^*$  is deterministic and  $(N, \tilde{\mathbf{g}}, \mathbf{a}; x)$  is fixed, the values  $\tilde{h}$  and  $\tilde{b}$  are fixed as well. Since we have  $(\tilde{h}, \tilde{b}; x) \notin L_{\text{EDL2}}[N, \tilde{\mathbf{g}}, \mathbf{a}]$  with a positive probability, we may deduce that  $\tilde{b}^2 \neq \tilde{h}^{2x}$ . We denote  $b := \tilde{b}^2, h := \tilde{h}^2$ , and  $y = \frac{b}{h^x}$ .
2. **Receiving proofs:**  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  calls  $\mathcal{P}_2^*$  and receives a (deterministic) claim  $\mathbf{u}, v$ .  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  then forks the protocol and sends at most  $\varepsilon^{-1}$  random challenges, until receiving the first valid response. If none of the responses is valid, then  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  outputs the failure symbol  $\perp$ . Otherwise, for one of the challenges  $e_1$  it received a valid response. In this case,  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  forks the protocol and sends the challenges  $e_1 + 1, e_1 + 2, \dots, \min(e_1 + \frac{2}{\varepsilon} - 1, 2^\kappa - 1)$ . If none of the responses to these challenges is valid, then  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  outputs  $\perp$ . Otherwise,  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  has two accepting transcripts  $(e_1, z_1)$  and  $(e_2, z_2)$ , such that  $e_1 < e_2 < e_1 + \frac{2}{\varepsilon}$ . Output  $(y, e_2 - e_1)$ .

*Proof (of Lemma 4.6).* Consider  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  depicted in Algorithm 4.2. The time complexity of  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  is dominated by that of  $\mathcal{P}^*$ , and is therefore bounded by  $\tilde{\mathcal{O}}\left(\frac{T}{\varepsilon}\right)$ . Let us prove that whenever  $\mathcal{A}_{\text{LO}}$  does not return  $\perp$ , its output is valid with high probability. Clearly,  $y \in \text{QR}_{N^2}$ ,  $e' \in (0, \frac{2}{\varepsilon})$ , and  $y \neq 1$ , so it remains to show that  $\text{ord}(y^{e'})$  is a  $\beta_{\sigma_0}$ -smooth number.

Since the transcripts  $(e_1, z_1)$  and  $(e_2, z_2)$  are both accepting, the following equations hold (modulo  $N^2$ ):

$$\mathbf{u} = \mathbf{g}^{z_1} \cdot \mathbf{a}^{e_1}, \quad v = h^{z_1} \cdot b^{e_1}, \quad \mathbf{u} = \mathbf{g}^{z_2} \cdot \mathbf{a}^{e_2}, \quad \text{and} \quad v = h^{z_2} \cdot b^{e_2}.$$

Denoting  $e' = e_2 - e_1$ ,  $z' = z_2 - z_1$ , we obtain

$$1 = \mathbf{g}^{z'} \cdot \mathbf{a}^{e'} \pmod{N^2} \quad \text{and} \quad 1 = h^{z'} \cdot b^{e'} \pmod{N^2}.$$

Since  $\mathbf{g}$  contains a  $\beta_{\sigma_0}$ -almost generator, we may pick  $i \leftarrow [c]$  and set  $g = \mathbf{g}_i$ ,  $a = \mathbf{a}_i$ , and deduce that  $g$  is a  $\beta_{\sigma_0}$ -almost generator with probability  $\geq \frac{1}{c}$ . By Lemma 4.7, there exists an element  $\eta \in \text{QR}_{N^2}$  such that  $\text{ord}(\eta)$  is  $\beta_{\sigma_0}$ -smooth and  $\langle g, \eta \rangle = \text{QR}_{N^2}$ . Therefore, there exist  $\alpha, \delta$  such that  $h = g^\alpha \eta^\delta$  and so

$$1 = g^{z'} \cdot a^{e'} \pmod{N^2} \quad \text{and} \quad 1 = g^{\alpha \cdot z'} \eta^{\delta \cdot z'} \cdot b^{e'} \pmod{N^2}.$$

Recall that  $\alpha, \delta$  and  $\eta$  are unknown to  $\mathcal{A}_{\text{LO}}$ . Dividing the second equation by the first equation raised to the power of  $\alpha$  we get

$$1 = \frac{g^{\alpha \cdot z'} \eta^{\delta \cdot z'} \cdot b^{e'}}{g^{\alpha \cdot z'} \cdot a^{\alpha \cdot e'}} = \eta^{\delta z'} \left( \frac{b}{a^\alpha} \right)^{e'} \pmod{N^2}. \quad (6)$$

Recall that  $a = g^x$ . We may deduce that

$$\frac{b}{a^\alpha} = \frac{b}{h^x} \cdot \frac{h^x}{a^\alpha} = y \cdot \frac{g^{\alpha x} \eta^{\delta x}}{g^{\alpha x}} = y \eta^{\delta x} \pmod{N^2}.$$

Then, substituting  $\frac{b}{a^\alpha} = y \eta^{\delta x}$  in Eq. (6), we obtain

$$1 = \eta^{\delta z'} (y \eta^{\delta x})^{e'} = \eta^{\delta(z' + x e')} y^{e'} \pmod{N^2}.$$

Finally, raising to the power of  $\text{ord}(\eta)$ , we conclude that

$$1 = \left( \eta^{\delta(z' + x e')} y^{e'} \pmod{N^2} \right)^{\text{ord}(\eta)} = y^{\text{ord}(\eta) e'} \pmod{N^2}.$$

Since  $\text{ord}(y^{e'}) \mid \text{ord}(\eta)$  We may conclude that  $\text{ord}(y^{e'})$  is  $\beta_{\sigma_0}$ -smooth.

It remains to bound the probability that  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  succeeds. With probability at least  $\frac{1}{c}$   $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  picks  $g = \mathbf{g}_i$  which is a  $\beta_{\sigma_0}$ -almost generator, otherwise we will assume that it fails. In the first part,  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  sends  $\varepsilon^{-1}$  challenges. Since  $\mathcal{P}^*$  is deterministic, we may define an *accepting challenge* to be a challenge  $e$  for which  $\mathcal{P}^*$  returns a valid response. Each of the  $\varepsilon^{-1}$  challenges is accepting with

probability at least  $\varepsilon$ , and the challenges are independent, so the probability that none of them is accepting is at most  $(1 - \varepsilon)^{\varepsilon^{-1}} \leq e^{-1} \approx 0.37$ . Otherwise,  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  receives  $e_1$  which is sampled uniformly from the set of the accepting challenges.

In the next step,  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  sends the challenges  $e_1 + 1, e_1 + 2, \dots, \min(e_1 + \frac{2}{\varepsilon} - 1, 2^\kappa - 1)$ , and fails if none of them is accepting. Let us denote by  $\ell$  the number of accepting challenges  $e_1$  for which all the challenges  $e_1 + 1, e_1 + 2, \dots, \min(e_1 + \frac{2}{\varepsilon} - 1, 2^\kappa - 1)$  are not accepting, and bound  $\ell$ : The distance between any such two accepting challenges is at least  $\frac{2}{\varepsilon}$ , and they all lie in  $[0, 2^\kappa)$ , so  $\frac{2}{\varepsilon}(\ell - 1) \leq 2^\kappa$ , implying that  $\ell \leq \frac{\varepsilon 2^\kappa}{2} + 1$ . There are at least  $\varepsilon 2^\kappa$  accepting challenges, so the probability that a uniform accepting transcript fails in this step is at most  $\frac{\ell}{\varepsilon 2^\kappa} \leq \frac{\varepsilon 2^\kappa / 2 + 1}{\varepsilon 2^\kappa} = \frac{1}{2} + \frac{1}{\varepsilon 2^\kappa} \leq \frac{3}{4}$ .

To conclude,  $\mathcal{A}_{\text{LO}}^{(\mathcal{P}^*)}$  succeeds with probability at least  $\frac{1}{c} \cdot \frac{1}{4}(1 - e^{-1}) \geq \frac{1}{8c}$ . ■

Before proving our main theorem, we need the following two results, related to reducing elements of small order into factoring (proofs are in Appendix G):

**Lemma 4.8.** *Let Factor be a factoring algorithm. There exists an algorithm Factor' that, given inputs  $N, x, e$  such that*

- $N = PQ$  is a conforming bi-prime; and
- $x \in \text{QR}_{N^2}$  satisfies  $x \not\equiv 1 \pmod{N^2}$ ,  $e \neq 0$ , but  $x^e \equiv 1 \pmod{N^2}$ ,

*outputs  $P, Q$  in time* $\text{time}(\text{Factor}'(N, x, e)) \leq \text{time}(\text{Factor}(e)) + \text{polylog}(N)$ .

**Lemma 4.9.** *There exists an algorithm Factor'', that given  $N, x, \beta$  as input, where  $N = PQ$  is a conforming bi-prime,  $1 \neq x \in \text{QR}_{N^2}$ , and  $\text{ord}(x)$  is  $\beta$ -smooth for some  $\beta < N$ , outputs  $P, Q$  in time* $\text{time}(\text{Factor}''(N, x, \beta)) \leq \beta \log^3(N)$ .

Notably, the reductions in Lemma 4.8 and Lemma 4.9 are polynomial with  $e$  and  $\beta$ . Nevertheless, they are only applied to  $e, \beta$  that are  $\text{poly}(\kappa)$  bounded. We now prove Theorem 4.5, which constitutes the main contribution of this work:

*Proof (Proof of Theorem 4.5).* Our proof is based on ideas that are derived from the proof of Theorem 3 in [BG11].

Let  $\mathcal{P}^*$  be such a prover. We denote by  $\mathcal{P}_\omega^* = (\mathcal{P}_{1,\omega}^*, \mathcal{P}_{2,\omega}^*)$  the deterministic prover obtained by fixing  $\mathcal{P}^*$ 's random tape to  $\omega$ . Then  $\mathcal{A}_{\text{NU}}$  works as follows:

1. Receive  $1^\kappa, N$  and  $i := \text{advice}(1^\kappa)$ .
2. Sample  $\omega$  and  $(\tilde{\mathbf{g}}, \mathbf{a}; x) \leftarrow \text{Setup}_2(1^\kappa, 1^\sigma, N)$ .
3. Call  $(y, e') \leftarrow \mathcal{A}_{\text{LO}}^{(\mathcal{P}_\omega^*)}(N, \tilde{\mathbf{g}}, \mathbf{a}, 2^{-i})$ . If  $\perp$  is received, output  $\perp$ .
4. If  $y^{e'} \equiv 1 \pmod{N^2}$ , factor  $N$  using Lemma 4.8.
5. Otherwise, factor  $N$  using Lemma 4.9.

First, we show that there exist an advice  $i \leq \kappa - 2$  such that  $\mathcal{A}_{\text{NU}}$  factors  $N$  with probability at least  $\frac{1}{8c}$ . Denote  $I_i = (2^{-i}, 2^{-i+1}]$ , and denote by  $\omega_\gamma$  the

random tape of the verifier  $\mathcal{V}$ . We further denote by  $P$  the random variable that for each  $(\omega, N, \tilde{\mathbf{g}}, \mathbf{a}, x)$ , returns:

$$\Pr_{\omega} \left[ \begin{array}{l} (\tilde{h}, \tilde{b}; x) \notin L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}] \\ (\mathcal{P}_2^*(x) \leftrightarrow \mathcal{V}(\tilde{h}, \tilde{b})) = 1 \end{array} \right],$$

where  $P$  is defined over the product distribution over  $\mathcal{P}^*$ 's random tape for  $\omega$ , and  $(N, \tilde{\mathbf{g}}, \mathbf{a}, x)$  is sampled from **Setup**, conditioned on it being successful (that is,  $N$  is a conforming bi-prime and  $\tilde{\mathbf{g}}$  contains a  $\beta_{\sigma_0}$  almost generator). Intuitively,  $P$  is the success probability of  $\mathcal{P}^*$  to pass verification, which may depend on its random tape and the setup.

Therefore, by definition,  $\mathbb{E}(P) \geq \varepsilon$ . We claim that there exists  $i \leq \kappa - 2$  such that  $\Pr[P \in I_i] \geq \frac{2^i \varepsilon}{4\kappa}$ .<sup>9</sup>

Let us assume by contradiction that no  $i$  satisfies this inequality. Then

$$\mathbb{E}(P) \leq \sum_{i=1}^{\kappa-2} \Pr[P \in I_i] \cdot 2^{-i+1} + \Pr[P \leq 2^{-\kappa+2}] \cdot 2^{-\kappa+2} < \varepsilon/2 + 2^{-\kappa+2} < \varepsilon,$$

since  $\varepsilon > 2^{-\kappa+3}$ , in contradiction. We may conclude that there exists a value  $i \leq \kappa - 2$  such that  $\Pr[P \geq 2^{-i}] \geq \Pr[P \in I_i] \geq \frac{2^i \varepsilon}{4\kappa}$ . Fix this  $i$  as the advice tape for  $\mathcal{A}_{\text{NU}}$ . Therefore, with probability  $\geq \frac{2^i \varepsilon}{4\kappa}$ ,  $2^{-i}$  is a correct input to  $\mathcal{A}_{\text{LO}}$ , and therefore by Lemma 4.6,  $\mathcal{A}_{\text{NU}}$  receives a correct output  $(y, e')$  with probability  $\geq \frac{2^i \varepsilon}{4\kappa} \times \frac{1}{8c}$ . Then by Lemma 4.8 and Lemma 4.9,  $\mathcal{A}_{\text{NU}}$  outputs a non-trivial factorization of  $N$  with this same probability  $\geq \frac{2^i \varepsilon}{32\kappa c}$ .

Next, we bound the expected run-time of  $\mathcal{A}_{\text{NU}}$ . By Lemma 4.6, the run-time of Line 3 is  $\text{time}(\mathcal{A}_{\text{LO}}(\dots, \varepsilon = 2^{-i})) = \tilde{O}(T2^i)$ . Then, the run-time of Lines 4–5 is  $\tilde{O}(\beta_{\sigma_0}) + \text{timeFactor}(e')$ , where the second term is bounded by  $\tilde{O}(\varepsilon^{-1/2})$ . However, these lines are entered only upon success of Line 3, and we must take this into consideration when considering the *expected* run-time of  $\mathcal{A}_{\text{NU}}$ .

If we denote by  $\varepsilon_{\text{NU}}$  the precise success probability of  $\mathcal{A}_{\text{NU}}$  (in particular,  $\geq \frac{2^i \varepsilon}{32\kappa c}$ ), the expected run-time of  $\mathcal{A}_{\text{NU}}$  is therefore  $T_{\text{NU}} \leq \tilde{O}(T2^i) + \varepsilon_{\text{NU}} \cdot \beta_{\sigma_0}$ , since Lines 4–5 are executed only upon success, to extract the factorization. Therefore,  $\frac{T_{\text{NU}}}{\varepsilon_{\text{NU}}} \leq \tilde{O}(32\kappa c \frac{T}{\varepsilon} + \beta_{\sigma_0})$ , as desired.

Finally, we observe that  $\text{advice}(\kappa)$  can be computed in time  $\tilde{O}(\frac{T}{\varepsilon^2})$ . This can be done by running  $\mathcal{A}_i$  for each  $i \leq \kappa - 2$  sequentially, for  $\mathcal{O}(2^i)$  sampled random tapes (for **Setup**<sub>1</sub>, **Setup**<sub>2</sub>,  $\mathcal{P}^*$ ), and observing the first  $i$  for which  $\mathcal{A}_{\text{LO}}$  does not output  $\perp$ . Hence, we can define a uniform adversary  $\mathcal{A}_{\text{U}}$  that computes  $\text{advice}(\kappa)$  and then calls  $\mathcal{A}_{\text{NU}}$ , yielding  $\text{time}(\mathcal{A}_{\text{U}})/\varepsilon_{\text{U}} \leq \tilde{O}(16\kappa c(\frac{T}{\varepsilon^2} + \beta_{\sigma_0}))$ . ■

<sup>9</sup> Note that if  $\mathcal{P}^*$  is PPT, we have that  $1/\varepsilon$  is  $\text{poly}(\kappa)$ , and  $i < \log_2(1/\varepsilon)$  (to get probability  $\leq 1$ ), and so  $2^i = \text{poly}(\kappa)$ .

#### 4.4 Batching

Batching allows a prover to convince a verifier of the correctness of many statements in an efficient way, i.e., much faster than it would take to prove (and verify) each statement separately. In the context of threshold decryption, batching may shift the bottleneck from the verification of the validity of a partial decryption to the combination of the parties' verified partial decryption into the plaintext.

Recall (from Protocol 4.1) that proving  $(\tilde{h}, \tilde{b}; x) \in \text{EDL}^2$  requires raising  $\mathbf{g}$  and  $h$  to the power of  $r$ , which is a large exponent. Then, without batching, proving  $B$  statements  $(\tilde{h}_i, \tilde{b}_i; x)$  requires performing  $2B$  exponentiations with large exponents: namely, raising  $\mathbf{g}$  and each  $h_i$  to the power of the corresponding large exponent  $r_i$ . To improve efficiency, we use the 'small exponent' (SE) technique, introduced in [BGR98] and followed by [APB+04]. The idea of the technique is to combine the  $(\tilde{h}_i, \tilde{b}_i)$  statements into a single statement  $(\tilde{h}, \tilde{b})$  using a random linear combination, such that  $\tilde{h} = \prod \tilde{h}_i^{t_i}$  and  $\tilde{b} = \prod \tilde{b}_i^{t_i}$ , and then use Protocol 4.1 only once, on the combined  $(\tilde{h}, \tilde{b})$ . Hence, raising to the power of a large exponent  $r$  happens only twice, just like in a proof of a single statement. The efficiency gain by that combination depends on the size of the coefficients  $t_i$ , which we show can be much smaller than the size of  $r$  without increasing the soundness error of the proof. The resulting proof of  $B$  statements requires both the prover and the verifier only  $2B$  small exponentiations and  $2$  large exponentiations ( $2B$  large exponentiations).

While our batching protocol is similar to existing protocols, our security analysis is novel. Similarly to Theorem 4.5, we use the notion of conforming bi-primes and present a reduction to the factoring problem, without assuming  $N$  is a safe bi-prime. Intuitively, the soundness of the batched protocol relies on the fact that it is not possible for the prover to pick statements  $(\tilde{h}_i, \tilde{b}_i; x)$ , of which at least one is incorrect, such that their random combination  $(\tilde{h}, \tilde{b}; x)$  is a correct statement (except for a negligible probability).

We note that using the small exponents technique requires the verifier to pick the coefficients  $t_i$  *only after* the prover committed to its statements, which incurs two additional rounds over Protocol 4.1. We show, however, that even this protocol (with five rounds) can be turned non-interactive using the Fiat-Shamir transform without significantly increasing soundness error (see Sect. 1).

Protocol 4.3 formally describes the batched proof of equality of discrete logs.

**PROTOCOL 4.3** ( $\Pi_{\text{EDL}^2}^B$ : *Batched Proof for EDL<sup>2</sup>*.)

**Inputs.**  $\mathcal{P}$  has  $(N, \tilde{\mathbf{g}}, \mathbf{a}; x)$  and  $(\tilde{h}_i, \tilde{b}_i)_{i \in [B]}$ , and  $\mathcal{V}$  has  $(N, \tilde{\mathbf{g}}, \mathbf{a})$  and  $(\tilde{h}_i, \tilde{b}_i)_{i \in [B]}$ , where  $(N, \tilde{\mathbf{g}}, \mathbf{a}; x) \leftarrow \text{Setup}(1^\kappa, 1^\sigma)$  and arbitrary  $(\tilde{h}_i, \tilde{b}_i)_{i \in [B]}$ .

**Protocol.**

1.  $\mathcal{V}$  checks that  $\tilde{h}_i, \tilde{b}_i \in \mathbb{Z}_{N^2}^*$ , and sends  $t_i \leftarrow [0, 2^\kappa)$  to  $\mathcal{P}$  for every  $i \in [B]$ .  
Then  $\mathcal{P}$  and  $\mathcal{V}$  compute  $\tilde{h} = \prod_{i \in [B]} \tilde{h}_i^{t_i}$ ,  $\tilde{b} = \prod_{i \in [B]} \tilde{b}_i^{t_i}$ .
2.  $\mathcal{P}$  and  $\mathcal{V}$  run  $\Pi_{\text{EDL}^2}$  (Protocol 4.1) to prove  $(\tilde{h}, \tilde{b}; x) \in L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}]$ .

Completeness follows by the fact that if  $\tilde{b}_i = \tilde{h}_i^x$  for all  $i \in [B]$  then  $\tilde{b} = \left(\prod_{i \in [B]} \tilde{b}_i^t\right) = \left(\prod_{i \in [B]} \tilde{h}_i^{tx}\right) = \left(\prod_{i \in [B]} \tilde{h}_i^t\right)^x = \tilde{h}^x$ , and so  $(\tilde{h}, \tilde{b}; x) \in \text{EDL}^2$ .

As for HVZK, we show that for every  $(\tilde{h}_i, \tilde{b}_i)_{i \in [B]}$  such that  $(\tilde{h}_i, \tilde{b}_i; x) \in L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}]$  for all  $i$ , there exists a PPT simulator  $\mathcal{S}$ , such that

$$\mathcal{S}_{(N, \tilde{\mathbf{g}}, \mathbf{a})}(\{\tilde{h}_i, \tilde{b}_i\}_{i \in [B]}) \stackrel{c}{=} \left\{ \text{View}(\mathcal{P}(\{\tilde{h}_i, \tilde{b}_i\}_{i \in [B]}; x) \leftrightarrow \mathcal{V}(\{\tilde{h}_i, \tilde{b}_i\}_{i \in [B]})) \right\}.$$

The simulator  $\mathcal{S}$  simply computes  $\tilde{h}$  and  $\tilde{b}$  as in the protocol, and runs the simulator associated with  $\Pi_{\text{EDL}^2}$  (Protocol 4.1) on  $(\tilde{h}, \tilde{b})$  and outputs  $(\mathbf{u}', v', e', z')$  as output by that simulator. The transcript produced by  $\mathcal{S}$  and the one under the real execution are statistically close with the exact same analysis as in the proof of Theorem 4.4. Next, we argue soundness:

**Theorem 4.10.** *Let  $\mathcal{P}^* = (\mathcal{P}_1^*, \mathcal{P}_2^*)$  be a stateful prover such that:*

$$\Pr_{\substack{(N, \tilde{\mathbf{g}}, \mathbf{a}; x) \leftarrow \text{Setup}(1^\kappa, 1^\sigma), \\ (\tilde{h}_i, \tilde{b}_i)_{i \in [B]} \leftarrow \mathcal{P}_1^*(x)}} \left[ \begin{array}{l} \exists i : (\tilde{h}_i, \tilde{b}_i; x) \notin L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}] \\ \left( \mathcal{P}_2^*(x) \leftrightarrow \mathcal{V}(\{\tilde{h}_i, \tilde{b}_i\}_{i \in [B]}) \right) = 1 \end{array} \middle| \begin{array}{l} N \text{ is a conforming} \\ bi\text{-prime and} \\ \tilde{\mathbf{g}} \text{ contains a } \beta_{\sigma_0} \\ \text{almost generator} \end{array} \right] \geq \varepsilon,$$

where  $\mathcal{P}_1^*, \mathcal{P}_2^*$  and  $\mathcal{V}$  receive  $(1^\kappa, 1^\sigma, N, \tilde{\mathbf{g}}, \mathbf{a})$  implicitly as inputs. Then assuming factorization is hard,  $\varepsilon = \text{neg}(\kappa)$ .

Notably, if  $\tilde{b}_i = -\tilde{h}_i^x$  for some  $i$ , then  $(\tilde{h}_i, \tilde{b}_i; x) \in L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}]$  (recall the definition of  $L_{\text{EDL}^2}$  from Sect. 4.1) and Protocol 4.3 may succeed. However, as mentioned earlier, decryption shares are squared before being used, so multiplying a decryption share by  $(-1)$  does not affect the decryption.

*Proof.* For brevity, denote the event that the prover attempts to cheat the verifier (assuming a successful setup) by

$$\text{Cheat} = \exists i \in [B] : (\tilde{h}_i, \tilde{b}_i; x) \notin L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}],$$

and the event that  $\mathcal{P}^*$  breaks soundness by

$$\text{Break} = [(\mathcal{P}_2^*(x) \leftrightarrow \mathcal{V}(\{\tilde{h}_i, \tilde{b}_i\}_{i \in [B]})) = 1] | \text{Cheat}.$$

Then, the theorem states that  $\varepsilon = \Pr[\text{Break}]$  is negligible, because:

$$\begin{aligned} \Pr[\text{Break}] &= \Pr \left[ \text{Break} | (\tilde{h}, \tilde{b}; x) \notin L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}] \right] \cdot \Pr[(\tilde{h}, \tilde{b}; x) \notin L_{\text{EDL}^2} | \text{Cheat}] \\ &\quad + \Pr \left[ \text{Break} | (\tilde{h}, \tilde{b}; x) \in L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}] \right] \cdot \Pr[(\tilde{h}, \tilde{b}; x) \in L_{\text{EDL}^2} | \text{Cheat}] \\ &\leq \Pr \left[ \text{Break} | (\tilde{h}, \tilde{b}; x) \notin L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}] \right] + \Pr[(\tilde{h}, \tilde{b}; x) \in L_{\text{EDL}^2} | \text{Cheat}]. \end{aligned}$$

Hence, we bound  $\Pr[\text{Break}]$  by the sum of  $\varepsilon_1 := \Pr \left[ \text{Break} | (\tilde{h}, \tilde{b}; x) \notin L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}] \right]$  and  $\varepsilon_2 := \Pr[(\tilde{h}, \tilde{b}; x) \in L_{\text{EDL}^2} | \text{Cheat}]$ . We have  $\varepsilon_1 = \text{neg}(\kappa)$  by Theorem 4.5 (otherwise we can construct an adversary  $\mathcal{P}^*$  who breaks the soundness of  $\Pi_{\text{EDL}^2}$  (Protocol 4.1)). In addition,  $\varepsilon_2 \leq 2^{-\kappa}$  by Lemma 4.11 below, assuming factorization is hard, which concludes the proof.  $\blacksquare$

**Lemma 4.11.** *Let  $\mathcal{P}_1^*$  be a PPT for which*

$$\Pr_{\substack{(N, \tilde{\mathbf{g}}, \mathbf{a}; x) \leftarrow \text{Setup}(1^\kappa, 1^\sigma), \\ (\tilde{h}_i, \tilde{b}_i)_{i \in [B]} \leftarrow \mathcal{P}_1^*(1^\kappa, 1^\sigma, N, \tilde{\mathbf{g}}, \mathbf{a}; x), \\ \{t_i\} \leftarrow [0, M]}} \left[ \begin{array}{l} \exists i : (\tilde{h}_i, \tilde{b}_i; x) \notin L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}] \\ \wedge (\tilde{h}, \tilde{b}; x) \in L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}] \end{array} \middle| N \text{ is a conforming bi-prime} \right] = \varepsilon,$$

where  $M \in \mathbb{N}$  is the coefficients domain,  $\tilde{h} = \prod_{i \in [B]} \tilde{h}_i^{t_i}$  and  $\tilde{b} = \prod_{i \in [B]} \tilde{b}_i^{t_i}$ . If  $\varepsilon \geq \frac{2}{M}$ , then there exist an algorithm that factors  $N$  in time  $\tilde{\mathcal{O}}(\varepsilon^{-0.5})$ .

*Proof.* Let  $(N, \tilde{\mathbf{g}}, \mathbf{a}; x) \leftarrow \text{Setup}(1^\kappa, 1^\sigma)$ , and let  $(\tilde{h}_i, \tilde{b}_i)_{i \in [B]}$  be the statements produced by  $\mathcal{P}_1^*$ . Assume that  $(\tilde{h}_{i_0}, \tilde{b}_{i_0}; x) \notin L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}]$  for some index  $i_0$  and that the probability of  $(\tilde{h}, \tilde{b}; x) \in L_{\text{EDL}^2}[N, \tilde{\mathbf{g}}, \mathbf{a}]$  is at least  $\varepsilon \geq \frac{2}{M}$ . Following the definition of  $L_{\text{EDL}^2}$ , we may denote  $h_i = \tilde{h}_i^2, b_i = \tilde{b}_i^2, h = \tilde{h}^2, b = \tilde{b}^2$ , and obtain that  $h_{i_0}^x \neq b_{i_0}$  and that the probability of  $h^x = b$  is at least  $\varepsilon \geq \frac{2}{M}$ . W.l.o.g, assume  $i_0 = 1$ . By definition,  $h^x = b$  if and only if  $(\prod_{i \in [B]} h_i^{t_i})^x = \prod_{i \in [B]} b_i^{t_i}$ , which is equivalent to  $\prod_{i \in [B]} \left(\frac{h_i^x}{b_i}\right)^{t_i} = 1$ . By isolating the  $i_0 = 1$  term we get:

$$\left(\frac{h_1^x}{b_1}\right)^{t_1} = \prod_{i \in [B] \setminus \{1\}} \left(\frac{h_i^x}{b_i}\right)^{-t_i}. \tag{7}$$

There exists a choice of  $t_2, \dots, t_B$ , for which Equation (7) holds with a probability of at least  $\varepsilon$  where  $t_1$  is uniformly distributed. Therefore, there exist at least  $\varepsilon M$  choices of  $t_1 \in [0, M]$  such that  $\left(\frac{h_1^x}{b_1}\right)^{t_1}$  yields the same result. Assuming  $\varepsilon \geq \frac{2}{M}$ , we may deduce that  $\text{ord}\left(\frac{h_1^x}{b_1}\right) \leq \left(\varepsilon - \frac{1}{M}\right)^{-1} \leq \frac{2}{\varepsilon}$ .

To conclude,  $\text{ord}\left(\frac{h_1^x}{b_1}\right) = \mathcal{O}(\varepsilon^{-1})$ . While  $\frac{h_1^x}{b_1}$  is known to  $\mathcal{A}$ , its order may not be known. Nevertheless,  $\mathcal{A}$  can find the order using Pollard’s rho algorithm in time  $\tilde{\mathcal{O}}(\varepsilon^{-0.5})$ . Next,  $\mathcal{A}$  has an element  $\frac{h_1^x}{b_1} \in \text{QR}_{N^2}$  with  $\frac{h_1^x}{b_1} \neq 1 \pmod{N^2}$ , and knows its order, which is  $\mathcal{O}(\varepsilon^{-1})$ . Therefore,  $\mathcal{A}$  can factor  $\text{ord}\left(\frac{h_1^x}{b_1}\right)$  (naively) in time  $\tilde{\mathcal{O}}(\varepsilon^{-0.5})$ . Therefore, by Lemma 4.8,  $\mathcal{A}$  can factor  $N$  in time  $\tilde{\mathcal{O}}(\varepsilon^{-0.5})$ . ■

### 4.5 Concrete Parameters

In Table 2 we refer to possible instantiations and compute the concrete statistical and computational security ensured by following the reductions.

### 4.6 Batch Verification

Batch verification allows a verifier to simultaneously verify proofs from multiple non-interactive provers, thereby reducing computational load. It is somewhat

**Table 2.** Possible parameter instantiations for Tiresias with  $\sigma_0 := 40$ . The computational security parameter is controlled by the complexity of factoring (by NIST) as a function of the modulus size  $\kappa(\log_2(N))$ . The non-uniform reduction in Theorem 4.5 suggests a multiplicative overhead of  $32c\kappa$ . Statistical security of the DKG is controlled by  $c$ , the number of  $\beta_{\sigma_0}$ -almost generator candidates.

Modulus Size	Number of $g_i$ 's	Computational Security	Statistical DKG Security
$\log_2(N)$	$c$	$\kappa - \log_2(32c\kappa)$	$\sigma = \sigma_0 c$
2048	1	$112 - 12 = 100$	40
3072	2	$128 - 13 = 115$	80

analogous to the batching procedure done using the ‘small exponent’ method, however since different provers have different verification keys  $a_j$ , instead of digesting multiple exponentiation operations into a single one, we get a multi-exponentiation [Pip80]. Essentially, instead of validating two (or more) equations  $\mathbf{g}^x = 1$  and  $h^y = 1$  separately, we may sample randomizers  $r_1, r_2 \in [0, 2^\kappa]$  and verify  $\mathbf{g}^{r_1 x} \cdot h^{r_2 y} = 1$ , and with  $1 - \text{neg}(\kappa)$  probability this implies the validity of both (or all) equations. An algorithm for computing  $\prod_{i=1}^C g_i^{t_i}$  is called a  $C$ -multi-exponentiation and can be computed more efficiently than performing  $C$  individual exponentiations (namely,  $g_i^{t_i}$ ) and then multiplying the results. Importantly, if batch verification fails, the verifier does not know the identity of the cheaters, since all claims were merged into one. However, the verifier can then verify each proof individually. Since the technique is known and security proof is analogous to that of batching, we refer to Appendix H for details.

## 5 Performance

We implemented our threshold Paillier scheme in Rust; the implementation is released as open source at <https://github.com/odsy-network/tiresias>. We use `crypto-bigint`<sup>10</sup> for constant-time computations over sensitive data to avoid leakage. In addition, we use `rayon`<sup>11</sup> for parallelism. Our evaluation demonstrates that the scheme can greatly leverage that.

We evaluate the performance of our scheme with number of parties,  $n$ , varying from 10 to 1000, and batch sizes,  $B$ , varying from 1 (without batching) to 1000. In cases where it applies, we use  $t = (2/3)n$  as the threshold. All experiments are conducted over two machine types: (1) AWS EC2 instance of type `c6i.24xlarge`<sup>12</sup> with 96 3<sup>rd</sup> generation Intel Xeon Scalable vCPUs @ 3.50GHz, and (2) MacBook Pro Apple M1 Max with 10 Cores @ 3.22GHz.

Our experiments use a 2048-bit bi-prime modulus  $N$  (equivalent to a 4096-bit Paillier modulus  $N^2$ ) where the decryption key  $d = \phi(N)[\phi(N)^{-1} \bmod N] \in \mathbb{Z}$

<sup>10</sup> <https://github.com/RustCrypto/crypto-bigint>.

<sup>11</sup> <https://github.com/rayon-rs/rayon>.

<sup>12</sup> <https://aws.amazon.com/ec2/instance-types/c6i/>.



is  $(t, n)$ -Shamir shared over  $\mathbb{Z}$  using tighter coefficient bounds as analysed in Appendix B.2 above. Presented run-times are averages over 10 runs. Appendix D reports concrete run-times in tables, below we present illustrative graphs only.

As presented in Table 1, existing protocols [FS01, HMR+19] that rely on standard hardness assumptions are similar to ours but require extensive repetitions. Therefore, we compare our performance with the “ideal” semi-honest (SH) model with no proofs. Comparison with [FS01] and [HMR+19] can be estimated by applying the multiplicative factors in Table 1 to the SH model’s benchmark.

*Figures & Tables.* In Fig. 1 and the supporting Table 3 we report the run-time for a single party to produce  $B$  decryption shares (for  $B$  different ciphertexts).

Then, in Fig. 2 and the supporting Table 5 we report the run-time for combining the decryption shares from the parties. In the malicious security model the parties also provide a proof of equality of discrete logs to prove the correctness of the decryption shares, in which case we use  $B$ -batched proofs, and the ‘Mal’ columns include the time it takes to verify these.

Finally, in Table 4 we isolate the time it takes to verify a  $B$ -batched proof.

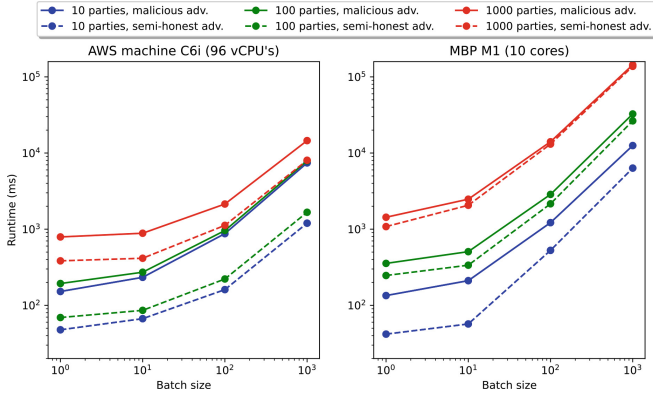
Apart for run-time, we report on the secret decryption key size and the proof size in bits, which both depend on the number of parties. For  $n = \{10, 100, 1000\}$  parties, the size in bits of the key  $d$  (which is shared over the integers) is  $\{4295, 5324, 19937\}$  and the proof size in bits is  $\{12743, 13772, 28385\}$ .

*Explaining the Results.* Note that in all figures and tables the number of parties,  $n$ , affects the run-time. This is due to the fact that  $n$  affects the party’s key-share size when shared over the integers (see Sect. 2.1 and Appendix B.2), which in turn affects both prover’s and verifier’s exponent size.

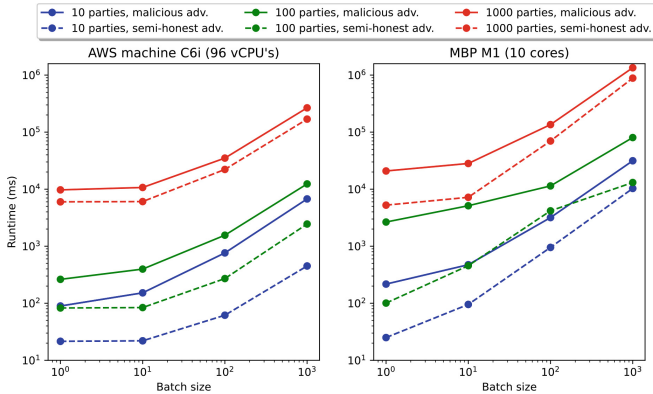
The attentive reader may observe an abrupt jump in run-time from a batch of  $B_1 = 100$  ciphertexts to a batch of  $B_2 = 1000$  in the C6i machine, and from  $B_1 = 10$  ciphertexts to  $B_2 = 100$  in the M1 machine. This jump is due to the parallelism of our implementation, which utilizes up to 96 cores of the C6i machine and up to 10 cores of the M1 machine. Up to  $B_1$  ciphertexts, the workload is quite concurrent and runs simultaneously for all ciphertexts in the batch, whereas above  $B_1$  ciphertexts work becomes sequential.

Importantly, the figures show that adding protection against a malicious adversary does not incur high overhead. For all  $n$  and  $B$  this overhead is a small constant and as  $n$  and  $B$  grow (toward  $n = 1000$  or  $B = 1000$ ) this constant reaches 1.5. We present the precise factors in the tables under the ‘ $\times$ ’ column.

The batching technique (along the multi exponentiation described in Section H.1) proves itself necessary if a truly scalable solution is needed. For decryption share computation (Table 3) with  $n = \{10, 100, 1000\}$ , the prover’s time for generating a decryption share for a *single* ciphertext is  $\{134, 355, 1434\}$  milliseconds on M1 machine, respectively. When generating decryption shares for  $B = 1000$  ciphertexts, the cost is only  $\{12.5, 32.5, 144\}$  milliseconds for each one, respectively, which is about  $10\times$  improvement. Similar results are obtained for combination of decryption shares (Table 2) and proof verification (Table 4).



**Fig. 1.** Time in milliseconds to generate a decryption share with and without proof of equality of discrete logs (e.g., in the presence of a malicious and semi-honest adversaries, respectively).



**Fig. 2.** Time in milliseconds to combine decryption shares from  $t = 2n/3$  parties of  $B$  ciphertexts, with and without proof of equality of discrete logs (e.g., in the presence of a malicious and semi-honest adversaries, respectively).

## 6 Conclusion

This paper introduces a novel security reduction technique, from the soundness of the proof of equality of discrete logs to the factoring problem. Combining our zero-knowledge proof (and its batching capabilities) with a large scale modulus generation (e.g., Diogenes [CHI+21]), we show for the first time, that threshold Paillier encryption scheme is practical under standard assumptions. In fact, we demonstrate that threshold Paillier is not only practical, but also ready for a large scale deployment with thousands of parties.

## References

- ACS02. Joy Algesheimer, Jan Camenisch, and Victor Shoup. [Efficient computation modulo a shared secret with application to the generation of shared safe-prime products](#). In *Advances in Cryptology-CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18-22, 2002 Proceedings 22*, pages 417–432. Springer, 2002.
- AFK22. Thomas Attema, Serge Fehr, and Michael Kloöß. [Fiat-shamir transformation of multi-round interactive proofs](#). In *Theory of Cryptography: 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part I*, pages 113–142. Springer, 2022.
- APB+04. Thomas Attema, Serge Fehr, and Michael Kloöß. [Fiat-shamir transformation of multi-round interactive proofs](#). In *Theory of Cryptography: 20th International Conference, TCC 2022, Chicago, IL, USA, November 7-10, 2022, Proceedings, Part I*, pages 113–142. Springer, 2022.
- BBBF18. Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. [Verifiable delay functions](#). In *Advances in Cryptology-CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part I*, pages 757–788. Springer, 2018.
- BDF+23. Jakob Burkhardt, Ivan Damgård, Tore Kasper Frederiksen, Satrajit Ghosh, and Claudio Orlandi. [Improved Distributed RSA Key Generation Using the Miller-Rabin Test](#). *Cryptology ePrint Archive*, 2023.
- BDO23. Lennart Braun, Ivan Damgård, and Claudio Orlandi. [Secure multi party computation from threshold encryption based on class groups](#). In *Annual International Cryptology Conference*, pages 613–645. Springer, 2023.
- BDTZ16. Carsten Baum, Ivan Damgård, Tomas Toft, and Rasmus Zakarias. [Better preprocessing for secure multiparty computation](#). In *Applied Cryptography and Network Security: 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings 14*, pages 327–345. Springer, 2016.
- BF97. Dan Boneh and Matthew Franklin. [Efficient generation of shared RSA keys](#). In *Advances in Cryptology-CRYPTO 97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17-21, 1997 Proceedings 17*, pages 425–439. Springer, 1997.
- BG11. Dan Boneh and Matthew Franklin. [Efficient generation of shared RSA keys](#). In *Advances in Cryptology-CRYPTO 97: 17th Annual International Cryptology Conference Santa Barbara, California, USA August 17-21, 1997 Proceedings 17*, pages 425–439. Springer, 1997.
- BGR98. Mihir Bellare, Juan A Garay, and Tal Rabin. [Fast batch verification for modular exponentiation and digital signatures](#). In *Advances in Cryptology-EUROCRYPT98: International Conference on the Theory and Application of Cryptographic Techniques Espoo, Finland, May 31-June 4, 1998 Proceedings 17*, pages 236–250. Springer, 1998.







- BS21. Omar Rafik Merad Boudia and Sidi Mohammed Senouci. [An Efficient and Secure Multidimensional Data Aggregation for Fog-Computing-Based Smart Grid](#). *IEEE Internet Things J.*, 2021.
- CCH+18. Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N Rothblum, and Ron D Rothblum. [Fiat-Shamir from simpler assumptions](#). *Cryptology ePrint Archive*, 2018.
- CHI+21. Megan Chen, Carmit Hazay, Yuval Ishai, Yuriy Kashnikov, Daniele Micciancio, Tarik Riviere, Abhi Shelat, Muthu Venkatasubramanian, and Ruihan Wang. [Diogenes: Lightweight scalable RSA modulus generation with a dishonest majority](#). In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 590–607. IEEE, 2021.
- CKY09. Jan Camenisch, Aggelos Kiayias, and Moti Yung. [On the Portability of Generalized Schnorr Proofs](#). In *EUROCRYPT*, volume 5479, pages 425–442. Springer, 2009.
- DdSGMRT21. Cyprien Delphe de Saint Guilhem, Eletheria Makri, Dragos Rotaru, and Titouan Tanguy. [The return of eratosthenes: Secure generation of rsa moduli using distributed sieving](#). In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 594–609, 2021.
- DJN10. Ivan Damgård, Mads Jurik, and Jesper Buus Nielsen. [A generalization of Paillier’s public-key system with applications to electronic voting](#). *International Journal of Information Security*, 9:371–385, 2010.
- DK01. Ivan Damgård and Maciej Koprowski. [Practical threshold RSA signatures without a trusted dealer](#). In *Advances in Cryptology-EUROCRYPT 2001: International Conference on the Theory and Application of Cryptographic Techniques Innsbruck, Austria, May 6-10, 2001 Proceedings 20*, pages 152–165. Springer, 2001.
- DN03. Ivan Damgård and Jesper Buus Nielsen. [Universally Composable Efficient Multiparty Computation from Threshold Homomorphic Encryption from Threshold Homomorphic Encryption](#). In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 247–264. Springer, 2003.
- DPSZ12. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. [Multiparty Computation from Somewhat Homomorphic Encryption](#). In *CRYPTO*, volume 7417, pages 643–662. Springer, 2012.
- FMM+23. Offir Friedman, Avichai Marmor, Dolev Mutzari, Yehonatan C Scaly, Yuval Spiizer, and Avishay Yanai. [Tiresias: Large scale, maliciously secure threshold paillier](#). *Cryptology ePrint Archive*, 2023.
- FPS01. Pierre-Alain Fouque, Guillaume Poupard, and Jacques Stern. [Sharing decryption in the context of voting or lotteries](#). In *Financial Cryptography: 4th International Conference, FC 2000 Anguilla, British West Indies, February 20-24, 2000 Proceedings 4*, pages 90–104. Springer, 2001.
- FS87. Amos Fiat and Adi Shamir. [How to prove yourself: Practical solutions to identification and signature problems](#). In *Advances in Cryptology-CRYPTO86: Proceedings 6*, pages 186–194. Springer, 1987.

- FS01. Pierre-Alain Fouque and Jacques Stern. [Fully distributed threshold RSA under standard assumptions](#). In *Advances in Cryptology-ASIACRYPT 2001: 7th International Conference on the Theory and Application of Cryptology and Information Security Gold Coast, Australia, December 9-13, 2001 Proceedings* 7, pages 310–330. Springer, 2001.
- GGN16. Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. [Threshold-optimal DSA/ECDSA signatures and an application to bitcoin wallet security](#). In *Applied Cryptography and Network Security: 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings* 14, pages 156–174. Springer, 2016.
- GO94. Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.
- HL23. Godfrey H Hardy and John E Littlewood. Some problems of ‘partitio numerorum’; iii: On the expression of a number as a sum of primes. *Acta Mathematica*, 44(1):1–70, 1923.
- HMR+19. Carmit Hazay, Gert Læssøe Mikkelsen, Tal Rabin, Tomas Toft, and Angelo Agatino Nicolosi. [Efficient RSA key generation and threshold paillier in the two-party setting](#). *Journal of Cryptology*, 32:265–323, 2019.
- KL14. Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. CRC Press, 2nd edition, 2014.
- KLM+20. Ralf Küsters, Julian Liedtke, Johannes Müller, Daniel Rausch, and Andreas Vogt. [Ordinos: A Verifiable Tally-Hiding E-Voting System](#). In *EuroS&P*, 2020.
- MT21. Dimitris Mouris and Nektarios Georgios Tsoutsos. Masquerade: Verifiable Multi-Party Aggregation with Secure Multiplicative Commitments. 2021.
- MV06. Hugh L. Montgomery and Robert C. Vaughan. [Multiplicative Number Theory I: Classical Theory](#). Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2006.
- NS10. Takashi Nishide and Kouichi Sakurai. [Distributed Paillier Crypto system without Trusted Dealer](#). In *Information Security Applications - 11th International Workshop, WISA 2010, Jeju Island, Korea, August 24-26, 2010, Revised Selected Papers*, volume 6513 of *Lecture Notes in Computer Science*, pages 44–60. Springer, 2010.
- Pai99. Pascal Paillier. [Public-key cryptosystems based on composite degree residuosity classes](#). In *Advances in Cryptology-EUROCRYPT 99: International Conference on the Theory and Application of Cryptographic Techniques Prague, Czech Republic, May 2-6, 1999 Proceedings* 18, pages 223–238. Springer, 1999.
- Pip80. Nicholas Pippenger. [On the evaluation of powers and monomials](#). *SIAM Journal on Computing*, 9(2):230–250, 1980.
- Pol74. John M Pollard. [Theorems on factorization and primality testing](#). In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 76, pages 521–528. Cambridge University Press, 1974.
- Rab98. Tal Rabin. [A Simplified Approach to Threshold and Proactive RSA](#). In *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 89–104. Springer, 1998.

- RSA78. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. [A method for obtaining digital signatures and public-key cryptosystems](#). *Communications of the ACM*, 1978.
- SB21. István András Seres and Péter Burcsi. [A note on low order assumptions in RSA groups](#). *Rad Hrvatske akademije znanosti i umjetnosti. Matematičke znanosti*, (546= 25):15–31, 2021.
- Sha79. Adi Shamir. [How to share a secret](#). *Communications of the ACM*, 22(11):612–613, 1979.
- Sho00. Victor Shoup. [Practical threshold signatures](#). In *Advances in Cryptology-EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19*, pages 207–220. Springer, 2000.
- VAS19. Thijs Veugen, Thomas Attema, and Gabriele Spini. [An implementation of the Paillier crypto system with threshold decryption without a trusted dealer](#). *ePrint*, 2019.



# Password-Protected Threshold Signatures

Stefan Dziembowski<sup>1,2</sup> , Stanislaw Jarecki<sup>3</sup> , Pawel Kedzior<sup>1</sup> ,  
Hugo Krawczyk<sup>4</sup> , Chan Nam Ngo<sup>5</sup> , and Jiayu Xu<sup>6</sup> 

<sup>1</sup> University of Warsaw, Warsaw, Poland  
{p.kedzior,s.dziembowski}@mimuw.edu.pl

<sup>2</sup> IDEAS NCBR, Warsaw, Poland

<sup>3</sup> University of California Irvine, Irvine, USA  
sjarecki@uci.com

<sup>4</sup> Amazon Web Services, Seattle, USA

<sup>5</sup> Privacy + Scaling Explorations, Ho Chi Minh City, Vietnam  
namcc@pse.dev

<sup>6</sup> Oregon State University, Corvallis, USA  
xujiay@oregonstate.edu

**Abstract.** We witness an increase in applications like cryptocurrency wallets, which involve users issuing signatures using private keys. To protect these keys from loss or compromise, users commonly outsource them to a custodial server. This creates a new point of failure, because compromise of such a server leaks the user's key, and if user authentication is implemented with a password then this password becomes open to an offline dictionary attack (ODA). A better solution is to secret-share the key among a set of servers, possibly including user's own device(s), and implement password authentication and signature computation using threshold cryptography.

We propose a notion of *augmented password-protected threshold signature* (aptSIG) scheme which captures the best possible security level for this setting. Using standard threshold cryptography techniques, i.e. threshold password authentication and threshold signatures, one can guarantee that compromising up to  $t$  out of  $n$  servers reveals no information on either the key or the password. However, we extend this with a novel property, that compromising *even all  $n$  servers* also does not leak any information, except via an unavoidable ODA attack, which reveals the key only if the attacker guesses the password.

We define aptSIG in the Universally Composable (UC) framework and show that it can be constructed very efficiently, using a black-box composition of any UC threshold signature [13] and a UC *augmented Password-Protected Secret Sharing* (aPPSS), which we define as an extension of prior notion of PPSS [30]. As concrete instantiations we obtain secure aptSIG schemes for ECDSA (in the case of  $t = n - 1$ ) and BLS signatures with very small overhead over the respective threshold signature.

Finally, we note that both the notion and our generic solution for augmented password-protected threshold signatures can be generalized to password-protecting MPC for any keyed functions.

## 1 Introduction

Threshold signatures have been studied for over 30 years [19]. Recently, their practical applicability increased significantly due to the use of signatures in blockchains and cryptocurrencies, especially for transaction authorization on behalf of users. In particular, multiple schemes have been developed for threshold ECDSA given the wide use of ECDSA in blockchains, e.g. [14, 20, 24, 33]. Recall that in a  $(t, n)$ -threshold signature the private signing key is shared between a set of  $n$  servers, and  $t + 1$  of them must collaborate to produce a signature; security requires that breaking into any  $t$  servers does not allow an attacker to forge signatures. Users that utilize a signature to authorize electronic transactions, e.g., the transfer of monies between accounts, but want to protect their keys from loss or compromise, can outsource signature generation to a trusted service that implements a threshold signature scheme. Yet, this setting raises the question of how a user can authorize the servers to sign on her behalf. An attacker who impersonates the user in this authorization process can request signatures on messages of its choice. On the other hand, if this authentication requires a user-held cryptographic key then we have a chicken-and-egg problem: we outsourced one user's key but we still require the user to hold another.

We can break this loop if we consider a setting where the authorization depends on a user's *password*. However, this presents another conundrum: Asking the user to pick an independent password for each server requires too much memorization (without secure storage), but using the same password with each server would create  $n$  points of failure, because an attacker who manages to break any one of the  $n$  servers would be able to run an offline dictionary attack against the user's password, and then use the password to authorize all other servers to sign any message.

**Augmented Password-Protected Threshold Signatures.** Our goal is a threshold signature scheme where all the user needs to authorize messages to be signed is a *single password*. The break of any  $t$  servers should leak no information that allows to attack either the signature scheme or the password, and the security should not rely on any secret or public keys stored or carried by the user. We refer to this notion as *Password-protected Threshold Signature* (ptSIG).

But we want more: We want that *even after the compromise of more than  $t$  servers (and possibly all  $n$  servers)*, the only information the attacker can gain requires finding the right password via an *exhaustive offline dictionary attack* (ODA). (Note that if a password triggers correct signature generation then an ODA on all-servers compromise is unavoidable.) In other words, the password should not only authenticate the user to the servers, but even if all servers are compromised they cannot produce signatures unless the attacker guesses the password. In particular, a solution that simply secret-shares the signing key among the servers would not work. In summary, we seek solutions that offer the following guarantees:

1. Each protocol execution, either by the user or by the servers, allows the attacker an online password test for only one password guess.



2. Compromising up to  $t$  servers results in no security loss, i.e. the attacker learns no information on either the signature key or the password.
3. Compromising  $t + 1$  or more servers (even all  $n$ ) does not give the attacker any information either, without the attacker first succeeding in an exhaustive offline dictionary attack (ODA) against the user’s password.

Properties 1 and 2 can be achieved by a composition of threshold Password-Authenticated Key Exchange (tPAKE) [35] and threshold signature scheme (tSIG) [18]. However, property 3 is not implied by such composition, and indeed does not seem easy to achieve using any tPAKE and tSIG schemes alone.

**Support for Server-Side Security Mechanisms.** We add one further requirement, and we refer to a notion which satisfies all requirements 1–4 as *Augmented Password-protected Threshold Signatures* (aptSIG):

4. An attacker who knows the password, can sign only one message per each interaction with  $t + 1$  servers, and only if these servers agree to sign it. In particular, if the attacker compromised  $t' \leq t$  servers, it can sign only one message per each interaction with  $(t + 1) - t'$  uncompromised servers.

Property 4 implies that the scheme cannot reveal the signing key to the user even if they hold the right password, as this would allow an attacker who compromises the password to sign messages without further server involvement. In contrast, an aptSIG scheme can limit such attacker by several mechanisms, such as *rate-limiting*, i.e. allowing only a limited number of signatures per time interval; implementing *multi-factor authentication*, which the attacker would need to bypass even if it learns the password; and signing messages only if they are compliant with an *application policy*, i.e. only messages with application-compliant semantics (e.g. including the correct current date). Note that Property 4 also protects the user in case of a break into the client machine: Such break might leak the password, but it cannot leak the signing key.

**Augmented Password-Protected Secret Sharing (aPPSS).** We introduce a protocol tool that plays an essential role in our aptSIG construction. Recall the notion of *Password-Protected Secret Sharing* (PPSS) [5]. A  $(t, n)$ -PPSS scheme allows user  $U$  to share a secret  $s$  among  $n$  servers and “protect” this sharing by a password  $\text{pw}$ , in the sense that PPSS reconstruction will recover  $s$  if and only if the user interacts with  $t + 1$  servers using the same password  $\text{pw}$ . (No extra user storage or authentication infrastructure such as PKI is assumed except during user registration.) PPSS security requires that compromising any  $t$  servers leaks no information on either the secret  $s$  or the password  $\text{pw}$ . However, for the purpose of building an aptSIG scheme, we need a stronger notion of PPSS with the following additional property: a compromise of more than  $t$  servers (even all  $n$  of them) still does not leak  $s$  and  $\text{pw}$  immediately, but only allows the attacker to stage an offline dictionary attack on the password, and this offline attack will leak  $s$  only if the attacker finds  $\text{pw}$ . We formalize this notion in the Universally Composable (UC) model [11] and refer to it as *augmented PPSS* (aPPSS), and we show that an existing PPSS scheme of [30] sufficiently realizes this stronger notion.

**From aPPSS to aptSIG.** Armed with the aPPSS tool, we build an aptSIG as follows. We start with a threshold signature scheme (tSIG) which relies on  $n$  servers and an additional entity  $U$ , called the user, where breaking the tSIG scheme requires breaking into  $t + 1$  servers *plus* compromising  $U$ . A tSIG scheme for this “1+threshold” access structure can be obtained from regular  $(t, n)$ -threshold signature by e.g. providing multiple shares to the user, but many threshold signatures can be adapted to this access structure more efficiently, as we exemplify by the BLS-based construction of Sect. 2.1.

At a high level, our aptSIG scheme works as follows:

- At initialization, which we assume runs over authenticated channels, e.g. using PKI for server authentication<sup>1</sup>, the tSIG scheme is initialized so that the servers and the user get the information needed to later run the signing protocol. Let  $\text{ts}_U$  denote the state that  $U$  needs to store to run tSIG signature protocol (this would include the share of the signature key, but also possibly the keys needed to authenticate/encrypt tSIG protocol messages). In addition, servers and  $U$  initialize an aPPSS instance under the user’s password which produces a random secret  $\text{sk}$  learned by  $U$ . The user authenticates-and-encrypts the state  $\text{ts}_U$  under key  $\text{sk}$  to obtain an authenticated encryption ciphertext  $\text{aec}_U$ , and sends  $\text{aec}_U$  to all servers who store it.  $U$  then erases all information and only remembers its password.
- To sign message  $m$ , party  $U$  and the servers run aPPSS reconstruction by which  $U$ , using its password, retrieves  $\text{sk}$ . The servers send  $\text{aec}_U$  back to  $U$  who authenticates-and-decrypts it under  $\text{sk}$  to learn its tSIG state  $\text{ts}_U$ . Finally, now that  $U$  holds its tSIG state,  $U$  and the servers run the tSIG scheme to sign  $m$ .

### Definitions, Generic Construction, Efficient aptSIG Instantiations.

Regarding the security of our construction, all of our constructions are defined in the UC model, which is essential for security under arbitrary composition: First, we frame the new notions of aPPSS and aptSIG as UC functionalities; second, we generalize the UC tSIG notion of Canetti et al. [13], which was defined only for the  $n$ -out-of- $n$  setting, to arbitrary  $(t, n)$ -threshold and 1+threshold access structures.

Next, we show how to efficiently realize our UC aptSIG notion: the schematic outline above provides a generic design of UC aptSIG scheme from any UC aPPSS and UC tSIG that supports the 1+threshold access structure. In this construction, the only overhead incurred while compiling a tSIG to an aptSIG is the cost of the aPPSS scheme, which can be instantiated efficiently: our UC aPPSS scheme, which is essentially identical to the PPSS of [30], is a generic construction from any UC Oblivious PRF (OPRF), and using the 2HashDH OPRF of [30] it requires only two communication flows and its computational cost is 1 exponentiation for each server and  $t + 2$  for the user.

<sup>1</sup> Authenticated channels between user and servers are needed at initialization in order for the user to identify the servers it is communicating with, but such channels, or PKI, are not needed for later signature generation.

At first glance, it seems that this generic construction leads to a UC-secure aptSIG implementation of ECDSA based on the UC ECDSA scheme of [13] adapted to the 1+threshold access structure. However, that scheme was shown secure only for the additive  $n$ -out-of- $n$  sharing, so the result in [13] only implies an aptSIG with  $t = n - 1$ . In the general case, one would have to carefully verify whether the generalization of ECDSA of [13] to the  $(t, n)$ -threshold and 1+threshold settings realizes the UC tSIG functionality for these access structures. Moreover, that scheme requires several rounds of interaction.

For the general case, we instead present a concrete round-minimal and highly practical aptSIG scheme (see Fig. 8 in Sect. 5) based on a threshold BLS signature [7, 8]. It requires only 2 communication flows in signing, 3 flows in initialization, uses no server-to-server communication, and takes  $O(1)$  exponentiations per server and  $O(n)$  exponentiations and bilinear maps for the user. We prove that this BLS-based scheme realizes the UC tSIG functionality for the 1+threshold access structure for any  $t \leq n$  s.t.  $\binom{n}{t}$  is polynomial in the security parameter; this probably can be extended to any parameters  $n, t$  using the results of Bacho and Loss [4] and Das and Ren [16] (see Sect. 2.1).

**Extensions to Password-Protected MPC.** While this paper develops definitions and mechanisms specific to the case of aptSIG, our approach and techniques can be generalized to provide “password-protection” of other cryptographic functions. For example, in the case of encryption, a user may want to decrypt encrypted data only in collaboration with a threshold of servers conditioned on knowledge of a password, and with additional assurances similar to those in our aptSIG treatment (e.g., enforcing a decryption policy by the servers, allowing for rate limits, etc.). In another example, one can consider a variant of aptSIG where the keyed function is a *blind* signature scheme, to keep messages signed hidden from the servers. In general, one can use this approach to password-protect multi-party computation of *arbitrary functions*, with security guarantees as in items 1–4 above, but with signatures replaced by an arbitrary keyed function. We leave such extensions and generalizations as subjects for future work.

**MPC for Obfuscated Point Function.** Finally, observe that aPPSS can be seen as a distributed computation of the point function

$$PF_{\text{pw},s}(x) = \begin{cases} s & \text{if } x = \text{pw} \\ \perp & \text{otherwise} \end{cases} .$$

The aPPSS protocol computes  $PF_{\text{pw},s}(\cdot)$  in a distributed setting, by user  $U$  holding input  $x$  and the servers holding the secret-sharing of the function description  $\langle \text{pw}, s \rangle$ , with  $U$  computing the output  $y = PF_{\text{pw},s}(x)$ . Moreover, the aPPSS property that even a compromise of all servers allows for recovery of  $s$  (and  $\text{pw}$ ) only via an offline dictionary attack, implies that the server-held shares reconstruct an *obfuscated* representation of point function  $PF_{\text{pw},s}$ , i.e. a software black-box which allows evaluation of  $PF_{\text{pw},s}(\cdot)$  on any input (e.g. password guess), but it leaks no information on  $(\text{pw}, s)$  unless one queries it on input  $x = \text{pw}$ . Thus, an

efficient aPPSS scheme implies an efficient evaluation of a *secret-shared obfuscated point function*, and as such it can find other applications.<sup>2</sup>

**Applications to Blockchain Wallets.** Some very attractive applications for threshold cryptography come from the blockchain domain. Recall that cryptocurrency coins are signature keys, spending a coin is implemented as a signing operation, and that storage of these signature keys is one of the most sensitive parts of the entire blockchain ecosystem. This problem is addressed by the use of so-called *hardware wallets* (see, e.g., [2]), *threshold wallets* (see, e.g., [15]), or *MPC wallets* (see, e.g., [3]). Our solution provides a stronger, practical, and flexible alternative to these methods. Our solution implements a threshold wallet, enabling storing cryptocurrencies in a threshold way, but it simultaneously protects them with a password in two ways: One way, which is standard, is that the user must use a correct password to access their cryptocurrency stored in a threshold wallet. The second way, which is novel, is that the shares stored by the threshold wallet parties are effectively encrypted under the password, so even corruption of all the threshold wallets parties does not leak the cryptocurrency keys in the clear. Instead, a corruption of all threshold wallet parties reveals an obfuscated “output-a-key-only-if-input-is-a-correct-password” black-box, which allows only offline dictionary attacks against a password, and leaks the cryptocurrency keys only if the adversary finds the correct password.

## 1.1 Further Related Works

**Threshold Signatures.** Threshold signatures were formalized by Desmedt and Frankel in [19] with precursors including [9, 17, 18]. Since then countless papers have studied threshold signatures for a variety of signature schemes. More recent work in the area has been motivated by cryptocurrency applications with particular focus on Threshold ECDSA, e.g. [14, 20, 24, 33] as a prevalent signature scheme used in these applications. Among these works, our paper adopts the UC formalism for threshold signatures from Canetti et al. [13] who present a threshold ECDSA scheme that realizes this formalism.

**Server-Aided Signatures.** Using passwords in the context of threshold signatures has been studied in the setting of server-aided signatures and their variants [10, 23, 27, 34, 39]. These papers address the case of a user with access to a dedicated device that stores a strong signing key but requires user’s password to generate signatures. The password prevents an attacker that gets hold of the device from producing signatures at will, but an attacker can run an offline dictionary attack by entering password guesses to the device. To prevent such dictionary attacks these works add a remote server with whom the device shares

---

<sup>2</sup> McQuoid et al. [36] made a related observation, that a (non-threshold) OPRF implements secure 2PC for evaluating (non-secret-shared) obfuscated point functions, and used it to construct 2PC on obfuscated inputs for a larger class of functions.

the signing key and whose participation is required for producing signatures. The user typically enters its password on the device, but the interaction with the remote server limits the number of password attempts an adversary can try once it controls the user’s device. Some of the schemes also support hiding the message being signed from the remote server. Most schemes in the literature consider a single remote server but e.g. the work of [39] includes distributing the remote server into a group of servers using a threshold signature scheme.

However, in all these cases, the user depends on its own device for generating signatures. In particular, the device stores strong cryptographic keys. Our setting is different. We assume users that carry with them nothing but their memorized passwords; they do not even carry high-entropy public values (such as servers’ public keys), let alone dedicated devices. In particular, in our solution, a user can trigger signatures by logging in from an arbitrary device.

**Password-Authenticated Threshold Signatures.** A different line of work that shares similarities with our paper, but targets a different application and has different security properties, is [1, 6]. These papers deal with a single sign-on setting where an identity provider (e.g., Google) authenticates users using passwords, and upon authentication provides users with signed tokens (which authenticates a user to some 3rd-party service). These works distribute the identity provider operation over a set of servers and use threshold cryptography in two ways: First, they use threshold password authentication (tPAKE) to authenticate users to the servers that implement a distributed identity provider; second, the servers use a threshold signature (tSIG) to sign the requested token.

However, in this application the signing key is the provider’s key, which is used to sign messages for *all* users, and it can be reconstructed if  $t + 1$  servers are compromised. By contrast, in our case each user shares its own private key across a set of servers, and neither this key nor the user’s password is leaked, except via offline dictionary attack, even if all servers collude. Indeed, none of the above cited works models or claims the “augmented” property we introduce in the aptSIG notion, namely that the break of the system requires not only that the attacker breaks into a sufficient threshold of servers, but that it also succeeds in subsequent exhaustive offline attack against the user’s password.

**Augmented Threshold PAKE and Proactive Security.** In a concurrent work, Gu et al. [28] define the notion of *augmented* threshold PAKE (atPAKE), where the term “augmented” denotes the same security property as in our augmented PPSS and augmented Password-protected Threshold Signatures. As the standard notion of tPAKE [35], a  $(t, n)$ -threshold atPAKE allows the user to authenticate using a password to a set of servers who secret-share password-related information, and the scheme leaks nothing if up to  $t$  out of  $n$  servers are compromised. However, if  $t+1$  or more servers are compromised, the password still doesn’t leak in the clear unless the attacker succeeds in an offline dictionary

attack (ODA). Intuitively, in atPAKE servers must secret-share a (salted) *hash* of the user’s password, rather than the password itself.

Apart from the fact that the work of [28] tackles a similar augmented property in the context of a different threshold cryptosystem (threshold PAKE rather than threshold password-protected signatures), their work also defines and constructs a UC threshold OPRF (tOPRF), and we believe that the tOPRF-to-PPSS compiler of [31] offers an alternative implementation of UC aPPSS. One reason this alternative aPPSS implementation is interesting is that all building blocks here can be made *proactively* secure: the tOPRF of [28] can be proactively secure, which leads to a proactively secure aPPSS, which (combined with a proactively secure threshold signature) in turn would result in a *proactively secure aptSIG*.

**Paper Organization.** Section 2 defines UC threshold signature (tSIG) for arbitrary access structures, and exemplifies it with a threshold BLS signature scheme. Section 3 defines Augmented Password-Protected Secret Sharing (aPPSS) and shows that the PPSS scheme of [30] realizes this notion. Section 4 defines Augmented Password-protected Threshold Signature (aptSIG), and shows a generic construction of secure aptSIG from aPPSS and tSIG schemes. Finally, in Sect. 5 we exemplify this generic compiler with an efficient and practical scheme based on threshold BLS.

Due to space constraints we defer some material to the full version of this paper [21]. Specifically, in the full version we include the proof of security for the threshold BLS scheme, we include the security proof for our aPPSS scheme, we compare our UC aPPSS model with prior PPSS definitions, we include the security proof for our aptSIG scheme, we introduce versions of our aptSIG model and the aptSIG protocol that add the property of *Perfect Forward Security* (PFS) to the basic model (here we sketch this extension in Sect. 4.1), and we show a concrete BLS-based instantiation of the PFS-aptSIG scheme.

## 2 Threshold Signatures

Figure 1 shows a generalization of the ideal functionality for threshold signature  $\mathcal{F}_{\text{tSIG}}$  of Canetti et al. [13] to an arbitrary access structure  $\mathbb{S}$ . The UC threshold signature model of [13] extends the formalization of standard (i.e. non-threshold) signatures as a UC functionality [12] (for prior and related work on UC signatures see references therein) to the *distributed* setting where the signing key is secret-shared among  $n$  servers. However, the UC formalization of [13] defined it solely for the case of an  $n$ -out-of- $n$  secret-sharing, where the signature is unforgeable if the adversary corrupts up to  $n-1$  servers, but all servers have to participate to issue a valid signature. Here we extend the definition of [13] to arbitrary access structures, including the  $(t, n)$ -threshold access structure the specialized “1+threshold” access structure we use in our aptSIG application.

Notation: We assume  $sid = (\dots, \mathbf{P})$  where  $\mathbf{P}$  is a list of parties, and we let  $\mathbf{P}_{sid}$  denote set  $\mathbf{P}$  specified by string  $sid$ .  $\mathbb{S}_{sid}$  denotes an access structure  $\mathbb{S}$  applied to set  $\mathbf{P}_{sid}$ , i.e. signatures for  $sid$  can be created only by a set  $A$  of parties s.t.  $A \in \mathbb{S}_{sid}$ . The functionality interacts with a set of parties  $\mathcal{P}$  and an adversary  $\mathcal{A}^*$ .  $\mathbf{Corr}$  is initialized to the initial set of corrupted parties.

**Key Generation:**

- [K.P] On  $(\text{tsig.keygen}, sid)$  from party  $P$  (or  $(\text{tsig.keygen}, sid, P)$  from  $\mathcal{A}^*$  if  $P \in \mathbf{Corr}$ ), record and send to  $\mathcal{A}^*$  tuple  $(\text{tsig.keygen}, sid, P)$ .
- [K.V] On  $(\text{tsig.publickey}, sid, V)$  from  $\mathcal{A}^*$ , if  $(\text{tsig.keygen}, sid, P)$  is recorded for all  $P \in \mathbf{P}_{sid}$  then record  $(sid, V)$ .
- [K.F] On  $(\text{tsig.keygencomplete}, sid, P)$  from  $\mathcal{A}^*$ , if  $\exists$  record  $(sid, V)$  then send  $(\text{tsig.publickey}, sid, V)$  to  $P$ .

**Signing:**

- [S.P] On  $(\text{tsig.sign}, sid, m)$  from  $P$  (or  $(\text{tsig.sign}, sid, P, m)$  from  $\mathcal{A}^*$  if  $P \in \mathbf{Corr}$ ), if  $\exists$  record  $(sid, V)$  then record and send to  $\mathcal{A}^*$  tuple  $(\text{tsig.sign}, sid, m, P)$ .
- [S.S] On  $(\text{tsig.signature}, sid, m, S, \sigma)$  from  $\mathcal{A}^*$ , if  $S \in \mathbb{S}_{sid}$  and tuple  $(\text{tsig.sign}, sid, m, P)$  is recorded for all  $P \in S$  then do the following:
  - [S.S.1] If  $\exists$  record  $(sid, m, \sigma, 0)$  then ignore this message;
  - [S.S.2] Else, if  $V(m, \sigma) = 1$  then record tuple  $(sid, m, \sigma, 1)$ ;
  - [S.S.3] If  $V(m, \sigma) = 0$  then ignore this message.
- [S.F] On  $(\text{tsig.signcomplete}, sid, m, P)$  from  $\mathcal{A}^*$ , if  $\exists$  record  $(sid, m, \sigma, 1)$  then send  $(\text{tsig.signature}, sid, m, \sigma)$  to  $P$ .

**Verification:**

- [V.V] On  $(\text{tsig.verify}, sid, m, \sigma, V)$  from  $P$ , send  $(\text{tsig.verify}, sid, m, \sigma, V)$  to  $\mathcal{A}^*$  and:
  - [V.1] If  $\exists$  records  $(sid, V)$  and  $(sid, m, \sigma, \beta')$  then set  $\beta := \beta'$ ;
  - [V.2] Else, if  $\exists$  record  $(sid, V)$  but no record  $(sid, m, \sigma', 1)$  for any  $\sigma'$  then set  $\beta := 0$ ;
  - [V.3] Else set  $\beta := V(m, \sigma)$ .
- [V.F] Record  $(sid, m, \sigma, \beta)$  and send  $(\text{tsig.verified}, sid, m, \sigma, \beta)$  to  $P$ .

**Party Compromise:** *(This query requires permission from the environment.)*

- [PC] On  $(\text{tsig.compromise}, sid, P)$  from  $\mathcal{A}^*$ , set  $\mathbf{Corr} := \mathbf{Corr} \cup \{P\}$ .

**Fig. 1.** Threshold signature functionality  $\mathcal{F}_{\text{tSIG}}$  for arbitrary access structure  $\mathbb{S}$

The threshold signature functionality  $\mathcal{F}_{\text{tSIG}}$  consists of three parts, Key Generation, Signing and Verification. In contrast to [13], our functionality omits Key-Refresh, but both versions support adaptive party compromise. Following [13], w.l.o.g. we identify a public key  $V$  with an arbitrary deterministic algorithm, i.e.

signature  $\sigma$  on message  $m$  is valid iff  $V(m, \sigma) = 1$ . Also following [13], we assume that if party  $P$  participates in key generation, then  $P$  runs on an instance identifier  $sid$  of a form  $\sigma = (\dots, \mathbf{P})$  where  $\mathbf{P}$  is a set of parties, including  $P$ , which  $P$  intends to involve in this instance. We denote the unique set  $\mathbf{P}$  specified by  $sid$  as  $\mathbf{P}_{sid}$ .

We use  $\mathbb{S}_{sid}$  to denote access structure  $\mathbb{S}$  instantiated over set  $\mathbf{P}_{sid}$ . For example, if  $\mathbf{P}_{sid} = \{P_1, P_2, P_3\}$  and  $\mathbb{S}$  is a 1-out-of-3 threshold access structure then  $\mathbb{S}_{sid} = \{\{P_1\}, \{P_2\}, \{P_3\}\}$ . Our aptSIG scheme in Sect. 4 relies on a threshold signature for a specialized “1+threshold” access structure  $\mathbb{S}$ , where  $\mathbf{P}_{sid}$  is a sequence of  $n + 1$  parties  $(P_0, P_1, \dots, P_n)$ ,  $P_0$  has a special status, and  $\mathbb{S}$  consists of all subsets  $S \subseteq \mathbf{P}_{sid}$  s.t. (1)  $P_0 \in S$  and (2)  $|S \cap \{P_1, \dots, P_n\}| \geq t + 1$ . In other words, a valid subset  $S$  must contain the special party  $P_0$  and at least  $t + 1$  of parties  $P_1, \dots, P_n$ . (Looking ahead, in our aptSIG implementation servers will play the role of parties  $P_1, \dots, P_n$ , and  $P_0$  will be the user.)

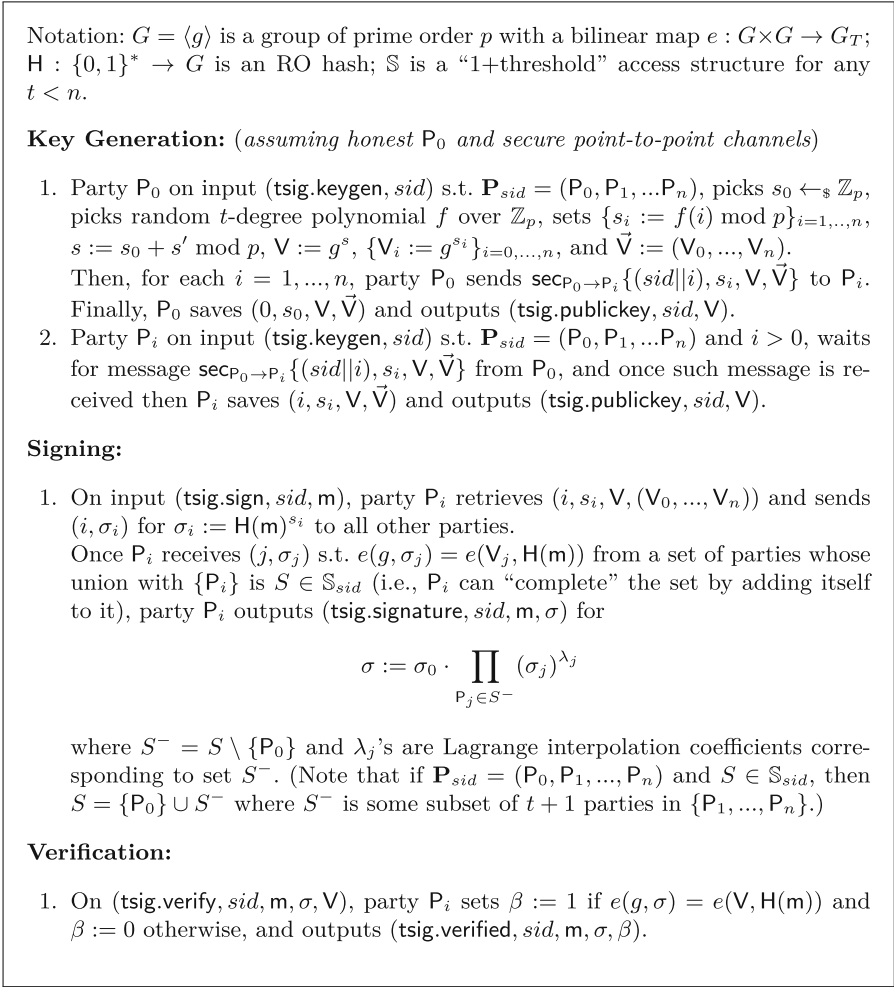
**Threshold Signature Functionality: Discussion.** To simplify notation in the key generation phase we assume that a signature scheme instance invoked with identifier  $sid$  generates a public key  $V$ , and a sharing of the corresponding private key, only if *all* parties in set  $\mathbf{P}_{sid}$  participate in the key generation using the same identifier  $sid$ . However, once the key generation succeeds, then a signature valid under the generated public key can be issued as long as it is requested by *any subset*  $S \subseteq \mathbf{P}_{sid}$  of parties s.t.  $S \in \mathbb{S}_{sid}$ .

Functionality  $\mathcal{F}_{\text{tSIG}}$  of Fig. 1 simplifies the one in [13] by omitting the option that lets all parties agree on a unique misbehaving party in each protocol phase. Supporting this option seems to require reliable authenticated broadcast, and since other protocols we use neither support a corresponding feature nor require reliable broadcast, we omit it here. Following [13], our functionality  $\mathcal{F}_{\text{tSIG}}$  does not support  $ssid$ 's in the signing phase and uses the message as an index of a signing protocol instance. Functionality  $\mathcal{F}_{\text{tSIG}}$  can be extended so every signer has additional input  $ssid$ , and signature is output only if for some subset  $S \in \mathbb{S}_{sid}$  all signers  $P \in S$  run on the same  $(ssid, m)$ . However, a cost-minimal protocol like the Threshold BLS scheme in Fig. 2 does not enforce such  $ssid$ -uniformity, so we opt for a simplified version of a signature functionality which, like the functionality of [13], doesn't enforce that either.

## 2.1 Threshold BLS Signature

The UC threshold signature functionality  $\mathcal{F}_{\text{tSIG}}$  can be implemented for BLS signature using the well-known protocol of Boldyreva [7]. Recall that a BLS signature [8] assumes a group  $G$  of prime order  $p$  with a bilinear map  $e : G \times G \rightarrow G_T$ , and defines  $\sigma$  as a signature on  $m$  under public key  $V = g^s$  if  $e(g, \sigma) = e(V, H(m))$ , where  $g$  generates  $G$  and  $H$  is a hash onto  $G$ . BLS signature is CMA-unforgeable in ROM under the *Gap DH* assumption, i.e. if the computational Diffie-Hellman is hard in  $G$  even on access to a DDH oracle [8].





**Fig. 2.** Threshold BLS scheme for the “1+threshold” access structure

Figure 2 shows a threshold BLS signature scheme that realizes functionality  $\mathcal{F}_{\text{tsig}}$  for the “1+threshold” access structure, for any threshold  $t < n$ . We support this access structure by combining a 2-out-of-2 sharing with a standard threshold sharing. Namely, sharing  $\vec{s} = (s_0, s_1, \dots, s_n)$  is formed by picking  $s_0 \leftarrow_{\mathbb{S}} \mathbb{Z}_p$ , setting  $(s_1, \dots, s_n)$  as a  $(t, n)$ -threshold secret-sharing of random  $s'$  in  $\mathbb{Z}_p$ , and setting the shared secret as  $s = s_0 + s' \bmod p$ . This way for any set  $S$

consisting of  $P_0$  and some  $t + 1$  parties in  $\{P_1, \dots, P_n\}$ , secret  $s$  can be reconstructed as  $s = s_0 + \sum_{P_i \in S^-} \lambda_i \cdot s_i \bmod p$  where  $S^- = S \setminus \{P_0\}$  and  $\lambda_i$ 's are Lagrange interpolation coefficients corresponding to set  $S^-$ .

**Standard Threshold Access Structure.** Note that setting  $s_0 = 0$  and removing  $P_0$  from signing transforms the protocol in Fig. 2 to a tSIG scheme which supports the standard  $(t, n)$ -threshold access structure. Moving in the other direction, we believe that most threshold signature schemes based on Shamir secret-sharing which realize  $\mathcal{F}_{\text{tSIG}}$  for the  $(t, n)$ -threshold access structure, can be transformed to support the “1+threshold” access structure using the above approach, but unfortunately it is not a black-box transformation and must be verified case by case.

**Distributed Key Generation.** The protocol in Fig. 2 realizes  $\mathcal{F}_{\text{tSIG}}$  in the presence of secure point-to-point channels in the Key Generation phase, and assuming that party  $P_0$  in list  $\mathbf{P}_{sid} = \{P_0, \dots, P_n\}$  is honest in that phase. The assumption on authenticated channels in key generation is unavoidable because  $\mathcal{F}_{\text{tSIG}}$  enforces that a shared key is generated only if all parties in  $\mathbf{P}_{sid}$  execute (`tsig.keygen, sid`), and using arbitrary key exchange protocol allows the participants to upgrade authenticated channels to secure point-to-point channels. As for the assumption on one honest party in key generation, this suffices for our aptSIG application, but this assumption can be easily eliminated by using any Distributed Key Generation (DKG) protocol for a discrete-log-based cryptosystem, e.g. [25, 38]. The analysis of the protocol in Fig. 2, presented in the full version of the paper [21], can be upgraded to this more general setting, e.g., by modeling the DKG subprotocol using the UC DKG functionality  $\mathcal{F}_{\text{DKG}}$  of Wikstrom [38], adapted to the 1+threshold access structure.

**Theorem 1.** *If BLS signature is CMA-unforgeable then the threshold signature scheme in Fig. 2 realizes functionality  $\mathcal{F}_{\text{tSIG}}$  for the “1+threshold” access structure for parameters  $t, n$  s.t.  $\binom{n}{t}$  is polynomial in the security parameter, assuming secure point-to-point channels and honest party  $P_0$  in the Key Generation phase.*

Proof of Theorem 1 is presented in the full version of the paper [21]:

**Security for Arbitrary  $t, n$  Parameters.** First, as sketched above, the scheme of Fig. 2 can be strengthened by replacing honest  $P_0$  with a secure DKG protocol. Moreover, Theorem 1 can be extended to arbitrary  $(t, n)$  values if the environment is restricted to *static corruptions*, i.e. all corruptions are made at the outset. This can be easily verified by inspecting the proof of Theorem 1 in the full version of the paper: the current reduction needs to guess a subset of corrupted parties, causing it to fail except with  $1/\binom{n}{t}$  probability; however, in the static corruption setting, the reduction no longer has to make such a guess.

Furthermore, Theorem 1 can be extended to arbitrary  $(t, n)$  values while allowing adaptive corruptions, following the analysis of threshold BLS by Bacho and Loss [4] in the Algebraic Group Model (AGM) [22], under the One-More Discrete Logarithm (OMDL) assumption. The analysis of [4] was done for the standard  $(t, n)$ -threshold BLS but we believe that it can be extended to BLS

which supports the 1+threshold access structure. The result of [4] also applies to several instantiations of a DKG protocol, including Pedersen’s JF-DKG [37] and New-DKG by Gennaro et al. [25]. In a recent work Das and Ren [16] showed a  $(t, n)$ -threshold BLS protocol which they show adaptively secure in the standard model, without AGM, and this protocol can also be extended to the 1+threshold setting. We note that the analysis of both [4] and [16] was arguing tSIG security defined via a game-based notion, so one also has to verify that they extend to the UC notion of tSIG captured by functionality  $\mathcal{F}_{\text{tSIG}}$ .

### 3 Augmented Password-Protected Secret Sharing

*Augmented Password-Protected Secret Sharing (aPPSS)* is a main component in our aptSIG scheme construction. Here we follow the informal description of aPPSS in the introduction with a formalization of this notion in the UC model. We then show how to instantiate this primitive with the PPSS construction of [29]. The latter masks shares of a threshold secret-sharing with outputs of Oblivious Pseudorandom Functions (OPRF) computed on the password. Since UC OPRF can be realized very inexpensively with protocol 2HashDH, this OPRF-based scheme leads to aPPSS with a retrieval cost of only 1 exponentiation per server and  $t + 2$  exponentiations per user. Concrete instantiation of aPPSS is shown in Fig. 8 as part of aptSIG protocol.

#### 3.1 Modeling Augmented Password-Protected Secret Sharing

The augmented PPSS functionality  $\mathcal{F}_{\text{aPPSS}}$  presented in Fig. 3 has four phases. In the *initialization* phase, user  $U$  can use command `ppss.unit` on input a password `pw` (`[L.U]`), to initialize a PPSS instance with a set of  $n$  servers whose identities  $\mathbf{P}_{\text{sid}} = \{P_1, \dots, P_n\}$  are assumed to be encoded in the session identifier, i.e.  $\text{sid} = (\text{sid}', \mathbf{P}_{\text{sid}})$ . The servers in  $\mathbf{P}_{\text{sid}}$  join this initialization using command `ppss.sinit` for matching  $\text{sid}$  and  $U$  (`[L.S]`). Finally, command `ppss.finit` from the ideal adversary  $\mathcal{A}^*$  corresponds to successful initialization, which allows  $U$  to output a secret random key `sk` which will be protected using this aPPSS instance (`[I.F]`). (Observe that this random key `sk` can be used to authenticate-and-encrypt arbitrary data, and indeed this is how we use it in the aptSIG protocol of Sect. 4).

The *reconstruction* command `ppss.urec` represents a user  $U'$  at a potentially different network entity, attempting to recover the secret key `sk` using password `pw'`, which may or may not be equal to `pw` used in initialization (`[R.U]`). The reconstruction operation is directed to a set of  $t + 1$  servers  $\mathbf{S}$ . We emphasize that the user maintains no state between the initialization and the reconstruction operations except for memorizing password `pw` and its username  $\text{sid}$  (although we also model the user forgetting `pw` and causing a failure during reconstruction—see below). In particular, the user might connect to a different set of servers in initialization and in reconstruction. Hence, for example, if a user executes the reconstruction protocol with a set of corrupted servers  $\mathbf{S}$ , the  $\mathcal{F}_{\text{aPPSS}}$  functionality guarantees that even in this case, the adversary can *only* perform an inevitable on-line guessing attack—which we explain below.

**Notation:** We assume strings  $sid$  of form  $sid = (\dots, \mathbf{P})$  where  $\mathbf{P} = (P_1, \dots, P_n)$ .  $\mathbf{P}_{sid}$  denotes set  $\mathbf{P}$  specified by string  $sid$ . The functionality interacts with a set of parties and an adversary  $\mathcal{A}^*$ . Let  $\mathbf{Corr}$  be the initial set of corrupted parties. Values  $t, n, \lambda$  are parameters. Functionality initializes  $\text{ppss.pwtested}(\text{pw}) := \emptyset$  for all  $\text{pw}$ , and  $\text{tx}(P_i) := 0$  for all  $P_i$ .

(The functionality code handles only one instance, tagged by a unique string  $sid$ .)

**Initialization:**

- [I.U] On  $(\text{ppss.uinit}, sid, \text{pw}, \text{sk}^*)$  from party  $U$  s.t.  $|\mathbf{P}_{sid}| = n$ : Send  $(\text{ppss.uinit}, sid, U)$  to  $\mathcal{A}^*$ . If  $U$  is honest then set  $\text{sk} \leftarrow_{\$} \{0, 1\}^\lambda$ , else set  $\text{sk} := \text{sk}^*$ . Save  $(\text{ppss.uinit}, sid, U, \text{pw}, \text{sk})$ . Ignore future  $\text{ppss.uinit}$  calls for same  $sid$ .
- [I.S] On  $(\text{ppss.sinit}, sid, i, U)$  from party  $S$ , or  $(\text{ppss.sinit}, sid, i, S, U)$  from  $\mathcal{A}^*$  for  $S \in \mathbf{Corr}$ , send  $(\text{ppss.sinit}, sid, i, S, U)$  to  $\mathcal{A}^*$ , save  $(\text{ppss.sinit}, sid, U, S, i)$ .
- [I.A] If  $\exists$  rec.  $(\text{ppss.uinit}, sid, U, \text{pw}, \text{sk})$  and  $(\text{ppss.sinit}, sid, U, S, i)$  s.t.  $S = \mathbf{P}_{sid}[i]$ , mark  $S$  as ACTIVE.
- [I.F] On  $(\text{ppss.finit}, sid)$  from  $\mathcal{A}^*$ , if  $\exists$  rec.  $(\text{ppss.uinit}, sid, U, \text{pw}, \text{sk})$  and all parties in list  $\mathbf{P}_{sid}$  are marked ACTIVE, send  $(\text{ppss.finit}, sid, \text{sk})$  to  $U$ .

**Server Compromise:** (This query requires permission from the environment.)

- [SC] On  $(\text{ppss.compromise}, sid, \mathbf{P})$  from  $\mathcal{A}^*$ , set  $\mathbf{Corr} := \mathbf{Corr} \cup \{\mathbf{P}\}$ .

**Reconstruction:**

- [R.U] On  $(\text{ppss.urec}, sid, ssid, \mathbf{S}, \text{pw}')$  from party  $U'$  or from  $U' = \mathcal{A}^*$ , send  $(\text{ppss.urec}, sid, ssid, U', \mathbf{S})$  to  $\mathcal{A}^*$ . If  $\exists$  record  $(\text{ppss.uinit}, sid, U, \text{pw}, \text{sk})$  then create record  $(\text{ppss.urec}, sid, ssid, U', \text{pw}, \text{pw}', \text{sk})$ , else create record  $(\text{ppss.urec}, sid, ssid, U', \perp, \text{pw}', \perp)$ . Ignore future  $\text{ppss.urec}$  calls for same  $ssid$ .
- [R.S] On  $(\text{ppss.srec}, sid, ssid, U')$  from party  $S$  or  $(\text{ppss.srec}, sid, ssid, S, U')$  from  $\mathcal{A}^*$  for  $S \in \mathbf{Corr}$ , send  $(\text{ppss.srec}, sid, ssid, S, U')$  to  $\mathcal{A}^*$ . If  $S$  is marked ACTIVE then increment  $\text{tx}(S)$  by 1.
- [R.F] On  $(\text{ppss.finrec}, sid, ssid, \mathbf{C}, \text{flag}, \text{pw}^*, \text{sk}^*)$  from  $\mathcal{A}^*$ , if  $\exists$  rec.  $(\text{ppss.urec}, sid, ssid, U', \text{pw}, \text{pw}', \text{sk})$  then erase it and send  $(\text{ppss.finrec}, sid, ssid, \text{sk}')$  to  $U'$  s.t.
  - [R.F.1] if  $\text{flag} = 1$ ,  $|\mathbf{C}| = t + 1$ , and  $\forall_{S \in \mathbf{C}}(\text{tx}(S) > 0)$  then set  $\text{tx}(S) \leftarrow$  for all  $S \in \mathbf{C}$ , and if  $\text{pw} = \text{pw}'$  then set  $\text{sk}' := \text{sk}$  else set  $\text{sk}' := \perp$ ;
  - [R.F.2] if  $\text{flag} = 2$  and  $\text{pw}^* = \text{pw}'$  then set  $\text{sk}' := \text{sk}^*$ ;
  - [R.F.3] otherwise set  $\text{sk}' := \perp$ .

**Password Test:**

- [PT] On  $(\text{ppss.testpw}, sid, S, \text{pw}^*)$  from  $\mathcal{A}^*$ , retrieve  $(\text{ppss.uinit}, sid, U, \text{pw}, \text{sk})$ . If  $\text{tx}(S) > 0$  then add  $S$  to set  $\text{ppss.pwtested}(\text{pw}^*)$  and set  $\text{tx}(S) \leftarrow$ . If  $|\text{ppss.pwtested}(\text{pw}^*)| = t + 1$  then return  $\text{sk}$  to  $\mathcal{A}^*$  if  $\text{pw}^* = \text{pw}$ , else return  $\perp$ .

**Fig. 3.** Augmented PPSS functionality  $\mathcal{F}_{\text{aPPSS}}$

Similar to the `ppss.uit` and `ppss.sinit` commands in the initialization phase, the `ppss.urec` and `ppss.srec` queries control resp. user and server entering into the reconstruction subprotocol. The crucial rule enforced by  $\mathcal{F}_{\text{aPPSS}}$  is that each server  $S \in \mathbf{P}_{\text{sid}}$  which joined the initialization is associated with a ticket counter  $\text{tx}(S)$ , and this ticket counter is incremented only if  $S$  enters into the aPPSS reconstruction instance. (Which in particular means that corrupt  $S$  can increase these tickets at will, see below.) Since we do not assume authenticated links,  $U'$  session can be “routed” by the adversary to arbitrary servers; hence in the `ppss.finrec` command,  $\mathcal{A}^*$  specifies a set  $\mathbf{C}$  of servers of its choice for participation in this reconstruction ([R.F]). The protocol finalization command `ppss.finit` can result in three possible outcomes:

- In a successful reconstruction session ([R.F.1]),  $U'$  outputs key  $\text{sk}$  created in the initialization, which can happen only if (I)  $\text{pw}' = \text{pw}$ , i.e.,  $U'$  runs on the correct password, (II)  $\text{tx}(S) > 0$  for all  $S \in \mathbf{C}$ , i.e., an adversary connected  $U'$  to servers who participated in the initialization and these servers engaged in PPSS reconstruction (note that each of these ticket is decremented at `ppss.finit`, hence each PPSS reconstruction can be “used” only once), and (III) the adversary allowed all these reconstructions to proceed without interference, which is modeled by setting  $\text{flag} = 1$ .
- The adversary can connect  $U'$  only to corrupt servers ([R.F.2]), which offers  $\mathcal{A}^*$  an ability to perform an on-line guessing attack on the user, because w.l.o.g. the adversary could execute the reconstruction protocol on behalf of corrupt servers on password  $\text{pw}^*$  and secret  $\text{sk}^*$  of its choice, and if  $\text{pw}^* = \text{pw}$  this would cause  $U'$  to reconstruct the adversarially chosen value  $\text{sk}^*$ . An on-line guessing attack is modeled by  $\mathcal{A}^*$  setting  $\text{flag} = 2$ .
- In all other cases the reconstruction fails and  $U'$  outputs  $\perp$  ([R.F.3]).

**Adaptive Compromise and Password Tests.** Command `ppss.compromise` allows  $\mathcal{A}^*$  to adaptively compromise any party  $P$  ([SC]). The only effect this has is if  $P = S$  for some  $S \in \mathbf{P}_{\text{sid}}$ , i.e. if  $\mathcal{A}^*$  compromises one of the servers participating in the initialization. Moreover, the effect of such compromise is not a leakage of any data (password  $\text{pw}$  or secret  $\text{sk}$ ), but an ability for  $\mathcal{A}^*$  to create unlimited “tickets” for  $\mathcal{A}^*$ , i.e. to increment  $\text{tx}(\mathcal{A}^*)$  at will. Such tickets can be used in the *test password* command `ppss.testpw` ([PT]): This query lets  $\mathcal{A}^*$  specify a password guess  $\text{pw}^*$  and a server  $S$ , and  $\mathcal{F}_{\text{aPPSS}}$  adds  $S$  to the set of servers for which  $\mathcal{A}^*$  tests  $\text{pw}^*$ , but each such action “costs” one ticket because  $\mathcal{F}_{\text{aPPSS}}$  decrements  $\text{tx}(S)$ . If the adversary tests the same  $\text{pw}^*$  on  $t + 1$  servers then if  $\text{pw}^* \neq \text{pw}$ ,  $\mathcal{F}_{\text{aPPSS}}$  responds  $\perp$ , but if  $\text{pw}^* = \text{pw}$  then  $\mathcal{F}_{\text{aPPSS}}$  leaks the aPPSS-protected secret  $\text{sk}$ . Note that the ticket-counting mechanism of  $\mathcal{F}_{\text{aPPSS}}$  enforces that any aPPSS instance completed by a server can be used either for a single instance of the honest user reconstructing a secret, or for a single instance of an adversary who uses `ppss.testpw` to attempt to reconstruct  $\text{sk}$  using a guessed password  $\text{pw}^*$ .

**On Authenticated Channels.** Functionality  $\mathcal{F}_{\text{aPPSS}}$  assumes authenticated channels during *initialization*: When user  $U$  specifies, via command `ppss.uit`, a set  $\mathbf{P}_{sid}$  of servers to initialize a secret-sharing instance, the adversary can only decide whether or not to allow this protocol to complete. This means that the adversary can block any party from communicating with the user, but it cannot divert this initialization to a different set of parties. In particular, only the corruption of parties in  $\mathbf{P}_{sid}$  may have an effect on the security of the protocol with consequences as described above. To enforce these conditions,  $U$  needs the means to authenticate each  $P \in \mathbf{P}_{sid}$  during initialization which is modeled via the authenticated channel functionality  $\mathcal{F}_{\text{AUTH}}$ . Importantly, we do not assume authenticated channels in the reconstruction phase of  $\mathcal{F}_{\text{aPPSS}}$ .

### 3.2 aPPSS Protocol

In Fig. 4 we show a UC aPPSS scheme, denoted  $\Pi_{\text{aPPSS}}$ , based on the PPSS scheme of Jarecki et al. [30]. Protocol  $\Pi_{\text{aPPSS}}$  uses UC OPRF, modeled by functionality  $\mathcal{F}_{\text{OPRF}}$ , and it assumes authenticated channels, modeled by functionality  $\mathcal{F}_{\text{AUTH}}$ , but it uses the latter only in the Initialization phase. At a high level the protocol proceeds as follows:

**Initialization:** User  $U$  asks for an OPRF evaluation  $\rho$  from each server  $S$ 's  $\mathcal{F}_{\text{OPRF}}$  using its password  $\text{pw}$ , and uses those evaluations as encryption keys for encrypting the threshold shares  $\{s_i\}$  generated with the Shamir's secret sharing scheme from a random secret  $s$ . Together with the user's password  $\text{pw}$ , and the encrypted shares  $\mathbf{e} = \{e_i\}$ ,  $U$  creates a cryptographic commitment  $[C||\text{sk}] = \text{H}(\text{pw}, \mathbf{e}, s)$  and uses  $\text{sk}$  as the secret key. The ciphertexts  $\mathbf{e} = \{e_i\}$  and  $C$  are then sent via the authenticated channel (via  $\mathcal{F}_{\text{AUTH}}$ ) and kept at the servers. The user keeps nothing besides remembering the password  $\text{pw}$ .

**Reconstruction:** To reconstruct, user  $U$  starts with asking for the OPRF evaluation  $\rho$  from each server  $S$ 's  $\mathcal{F}_{\text{OPRF}}$  using its password  $\text{pw}$  along with the ciphertexts  $\mathbf{e} = \{e_i\}$  and commitment  $C$ . The OPRF evaluations  $\{\rho_i\}$  are used to decrypt the ciphertexts to Shamir's shares  $\{s_i\}$  which can be used to reconstruct the secret  $s$  via interpolation. Finally the user  $U$  can recreate  $[C||\text{sk}] = \text{H}(\text{pw}, \mathbf{e}, s)$  and obtain  $\text{sk}$ , after checking that  $C$  matches the ones sent by the servers.

Protocol  $\Pi_{\text{aPPSS}}$  in Fig. 4 is, up to some small differences (e.g., using a global OPRF functionality) the same as the PPSS of Jarecki et al. [30], except that we replace generic non-malleable commitment used in [30] with a specific RO-based implementation  $\text{H}$ . However, the novelty here with respect to the PPSS protocol of [30] is its analysis as an *augmented* PPSS.

**Theorem 2.** *If  $\text{H}$  is a random oracle, then the protocol in Fig. 4 UC-realizes the  $\mathcal{F}_{\text{aPPSS}}$  functionality assuming access to the OPRF functionality  $\mathcal{F}_{\text{OPRF}}$  and the message authentication functionality  $\mathcal{F}_{\text{AUTH}}$ .*

Proof of Theorem 2 is shown in the full version of the paper [21].

Public parameters: Security parameter  $\lambda$ , threshold parameters  $t, n \in \mathbb{N}$  with  $t \leq n$ , field  $\mathbb{F} := \mathbb{GF}(2^\lambda)$ , hash function  $H$  with range  $\{0, 1\}^{2^\lambda}$ .

**Initialization for user U:**

1. On input  $(\text{ppss.uinit}, \text{sid}, \text{pw})$  s.t.  $|\mathbf{P}_{\text{sid}}| = n$ , send  $(\text{opr.f.eval}, [\text{sid}||i||0], \text{pw}, \mathbf{P}_{\text{sid}}[i])$  to  $\mathcal{F}_{\text{OPRF}}$  for each  $i \in [n]$ .
2. Wait for messages  $(\text{opr.f.eval}, [\text{sid}||i||0], \rho_i, \text{tr}_i)$  from  $\mathcal{F}_{\text{OPRF}}$  and  $(\text{sent}, [\text{sid}||i||0], \mathbf{P}_{\text{sid}}[i], \text{U}, \text{tr}'_i)$  from  $\mathcal{F}_{\text{AUTH}}$ , for all  $i \in [n]$ . Abort if  $\exists i \in [n]$  s.t.  $\text{tr}'_i \neq \text{tr}_i$ .
3. Pick  $s \leftarrow_{\S} \mathbb{F}$ , set  $(s_1, \dots, s_n)$  as a  $(t, n)$  Shamir secret sharing of  $s$  over  $\mathbb{F}$ .
4. Set  $e_i := s_i \oplus \rho_i$  for  $i \in [n]$ , set  $\mathbf{e} := (e_1, \dots, e_n)$ , set  $[C||\text{sk}] := H(\text{pw}, \mathbf{e}, s)$  s.t.  $|C| = |\text{sk}| = \lambda$ . Set  $\omega := (\mathbf{e}, C)$ .
5. Send  $(\text{send}, [\text{sid}||i||1], \mathbf{P}_{\text{sid}}[i], \omega)$  to  $\mathcal{F}_{\text{AUTH}}$  for each  $i \in [n]$  and output  $(\text{ppss.fininit}, \text{sid}, \text{sk})$ .

**Initialization for server S:**

1. On input  $(\text{ppss.sinit}, \text{sid}, i, \text{U})$ , send  $(\text{opr.f.init}, [S||\text{sid}])$  and  $(\text{opr.f.sndrcomplete}, [S||\text{sid}], 0)$  to  $\mathcal{F}_{\text{OPRF}}$ .
2. Given response  $(\text{opr.f.sndrtrans}, [S||\text{sid}], 0, \text{tr}_S)$  from  $\mathcal{F}_{\text{OPRF}}$ , send  $(\text{send}, [\text{sid}||i||0], \text{U}, \text{tr}_S)$  to  $\mathcal{F}_{\text{AUTH}}$ .
3. On  $(\text{sent}, [\text{sid}||i||1], \text{U}, \mathbf{P}_i, \omega)$  from  $\mathcal{F}_{\text{AUTH}}$ , save  $(\text{sid}, i, \omega)$ .

**Reconstruction for user U:**

1. On input  $(\text{ppss.urec}, \text{sid}, \text{ssid}, \mathbf{S}, \text{pw}')$  s.t.  $|\mathbf{S}| = t+1$ , send  $(\text{opr.f.eval}, [\text{sid}||j||\text{ssid}], \mathbf{S}[j], \text{pw}')$  to  $\mathcal{F}_{\text{OPRF}}$  for  $j \in [t+1]$ .
2. Wait for messages  $(\text{opr.f.eval}, [\text{sid}||j||\text{ssid}], \phi_j, \text{tr}_j)$  from  $\mathcal{F}_{\text{OPRF}}$  and messages  $(i_j, \omega_j)$  from  $\mathbf{S}[j]$ , for all  $j \in [t+1]$ . If  $\exists j_1 \neq j_2$  s.t.  $i_{j_1} = i_{j_2}$  or  $\omega_{j_1} \neq \omega_{j_2}$  or  $\exists j$  s.t.  $i_j \notin [n]$  (i.e., if  $i_j$ 's are not all distinct, or  $\omega_j$ 's are not all the same, or some  $i_j$  is out of range  $[n]$ ), output  $(\text{ppss.urec}, \text{sid}, \text{ssid}, \perp)$  and halt. Otherwise set  $\rho'_{i_j} := \phi_j$  for  $j \in [t+1]$  and  $I := \{i_j \mid j \in [t+1]\}$ .
3. Parse any  $\omega_j$  as  $(\mathbf{e}', C')$ , parse  $\mathbf{e}'$  as  $(e'_1, \dots, e'_n)$ , set  $s'_i := e'_i \oplus \rho'_{i_j}$  for all  $i \in I$ .
4. Interpolate  $\{(i, s'_i)\}_{i \in I}$  to recover secret  $s'$  and shares  $\{s'_i\}_{i \notin I}$ .
5. Set  $[C''||\text{sk}'] := H(\text{pw}', \mathbf{e}', s')$ . If  $C' \neq C''$  then reset  $\text{sk}' := \perp$ .
6. Output  $(\text{ppss.finrec}, \text{sid}, \text{ssid}, \text{sk}')$ .

**Reconstruction for server S:**

1. On input  $(\text{ppss.srec}, \text{sid}, \text{ssid}, \text{U})$ , retrieve record  $(\text{sid}, i, \omega)$  (if no such record then abort), send  $(\text{opr.f.sndrcomplete}, [S||\text{sid}], \text{ssid})$  to  $\mathcal{F}_{\text{OPRF}}$  and  $(i, \omega)$  to  $\text{U}$ .

**Fig. 4.** Protocol  $\Pi_{\text{APPSS}}$  which realizes  $\mathcal{F}_{\text{APPSS}}$  in  $(\mathcal{F}_{\text{OPRF}}, \mathcal{F}_{\text{AUTH}})$ -hybrid world

## 4 Augmented Password-Protected Threshold Signature

We introduce our model for Augmented Password-protected Threshold Signature (aptSIG), and we show a secure construction of aptSIG scheme by generic composition of aPPSS and a Threshold Signature (tSIG).

### 4.1 Modeling Augmented Password-Protected Threshold Signature

We model Augmented Password-protected Threshold Signature (aptSIG) using an ideal functionality  $\mathcal{F}_{\text{aptSIG}}$ , shown in Fig. 5 and Fig. 6. A  $(t, n)$ -threshold aptSIG involves  $n + 1$  parties, a user  $U$  and  $n$  server  $S_1, \dots, S_n$ , and it supports two distributed protocols, *initialization* and *signing*. An initialization protocol generates a public key for a signature scheme and protects the corresponding private key by secret-sharing it and protecting this sharing using user's password  $\text{pw}$  s.t. the sharing can be reconstructed only using this password. The signing protocol allows the user and the servers to sign any message  $m$  as long as (a) the user and at least  $t + 1$  of the servers agree to sign it, and (b) the user provides a matching password  $\text{pw}' = \text{pw}$  into the signing protocol. Therefore, aptSIG scheme functions as an *outsourced signature service* for party  $U$ , where  $U$ 's secret key is *distributed and password-protected* by the servers, but using the right password lets  $U$  obtain signatures as long as  $t + 1$  servers agree to sign.

Corruption of up to  $t$  out of  $n$  servers gives no information to the attacker, while corruption of  $t + 1$  or more servers allows the attacker to reconstruct only password-protected data. In particular, the data collected from all servers allows the attacker an *offline dictionary attack* against the password, but that is *all that it allows*. If the attacker finds the password via this offline search then security is gone, and in our scheme the attacker reconstructs the signature private key, but if the password is chosen with high-enough entropy and the dictionary attack fails then the attacker gets no information about the signature key even if it corrupts all  $n$  servers. We stress that in a secure aptSIG scheme the signing key can never be reconstructed in one place. In particular, if the password leaks but the adversary compromises fewer than  $t + 1$  servers then signatures can only be created via the on-line signing protocol. Consequently, servers  $S_i$  can function as *rate limiters* or *policy limiters*, i.e. they can apply whatever policy the environment specifies regarding the messages they can sign.

**Ideal Functionality  $\mathcal{F}_{\text{aptSIG}}$ .** In what follows we explain the security properties imposed by the ideal functionality  $\mathcal{F}_{\text{aptSIG}}$  of Fig. 5 and Fig. 6. Since we show that our aptSIG protocol of Sect. 4.2 securely realizes this functionality, this will in particular imply the security properties of that aptSIG scheme.

(1)  $\mathcal{F}_{\text{aptSIG}}$ : **Honest Party Operation.** Query  $(\text{ptsig.uit}, \text{sid}, \text{pw})$  from  $U$  models user  $U$  starting initialization on a password  $\text{pw}$  with  $n$  servers specified in identifier  $\text{sid}$ . (Using the convention of aPPSS, we assume  $\text{sid} = (\text{sid}', \mathbf{P}_{\text{sid}})$  for  $\mathbf{P}_{\text{sid}} = (S_1, \dots, S_n)$ .) Query  $(\text{ptsig.uit}, \text{sid}, i, U)$  from  $S \in \mathbf{P}_{\text{sid}}$  models server  $S$  entering into an initialization protocol, as an  $i$ -th server in list  $\mathbf{P}_{\text{sid}}$ , with  $U$  as an intended “owner” of this password-protected signature instance. Query



**Notation:** (This figure uses the same notation as in  $\mathcal{F}_{\text{aPPSS}}$ , see Figure 3)

**Initialization:**

- [I.U] On  $(\text{ptsig.uinit}, \text{sid}, \text{pw})$  from party  $U$  for  $\text{sid} = (\dots, \mathbf{P}_{\text{sid}})$  s.t.  $|\mathbf{P}_{\text{sid}}| = n$ , send  $(\text{ptsig.uinit}, \text{sid}, U)$  to  $\mathcal{A}^*$ , save  $(\text{sid}, U, \mathbf{P}_{\text{sid}}, \text{pw})$  and set flag  $\text{flag}_{\text{sid}} = 0$ . Ignore further  $\text{ptsig.uinit}$  calls for same  $\text{sid}$ .
- [I.S] On  $(\text{ptsig.sinit}, \text{sid}, i, U)$  from party  $S$ , or  $(\text{ptsig.sinit}, \text{sid}, i, S, U)$  from  $\mathcal{A}^*$  for  $S \in \text{Corr}$ , send  $(\text{ptsig.sinit}, \text{sid}, i, S, U)$  to  $\mathcal{A}^*$ , save  $(\text{sid}, U, S, i)$ .
- [I.F] On  $(\text{ptsig.uinit}, \text{sid}, V)$  from  $\mathcal{A}^*$ , if  $\exists$  record  $(\text{sid}, U, \mathbf{P}_{\text{sid}}, \text{pw})$  and records  $(\text{sid}, U, S, i)$  for each  $S \in \mathbf{P}_{\text{sid}}$ , then create record  $(\text{sid}, \mathbf{P}_{\text{sid}}, \text{pw}, V)$  and send  $(\text{ptsig.verificationkey}, \text{sid}, V)$  to  $U$ .

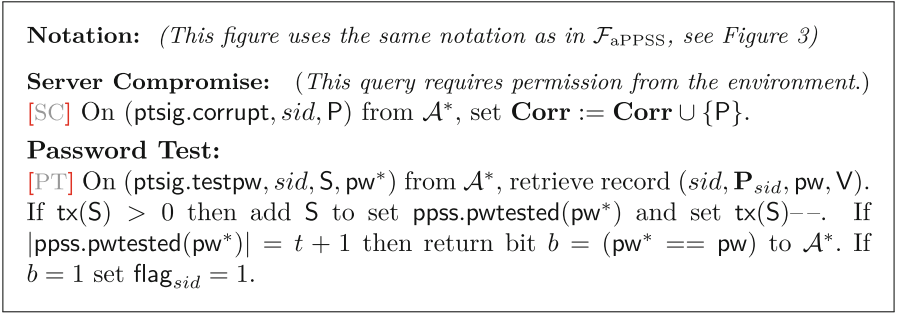
**Signing:**

- [S.U] On  $(\text{ptsig.usign}, \text{sid}, \text{ssid}, S, \text{pw}', m)$  from party  $U'$  or from  $U' = \mathcal{A}^*$ , send  $(\text{ptsig.usign}, \text{sid}, \text{ssid}, U', S, m)$  to  $\mathcal{A}^*$ . If  $\exists$  record  $(\text{sid}, \mathbf{P}_{\text{sid}}, \text{pw}, V)$  then save  $(\text{sid}, \text{ssid}, U', \mathbf{P}_{\text{sid}}, \text{pw}, \text{pw}', V, m)$ , else save  $(\text{sid}, \text{ssid}, U', \perp, \perp, \text{pw}', \perp, m)$ . Ignore further  $\text{ptsig.usign}$  calls for same  $\text{ssid}$ .
- [S.S] On  $(\text{ptsig.ssign}, \text{sid}, \text{ssid}, U', m)$  from party  $S$  or  $(\text{ptsig.ssign}, \text{sid}, \text{ssid}, S, U', m, b)$  from  $\mathcal{A}^*$  for  $S \in \text{Corr}$ , if  $\exists$  record  $(\text{sid}, \mathbf{P}_{\text{sid}}, \text{pw}, V)$  s.t.  $S \in \mathbf{P}_{\text{sid}}$  then send  $(\text{ptsig.ssign}, \text{sid}, \text{ssid}, S, U', m)$  to  $\mathcal{A}^*$ , save  $(\text{sid}, m, S)$ , set  $\text{tx}(S)++$  if  $S$  is honest or  $b = 1$ .
- [S.P] On  $(\text{ptsig.pretest}, \text{sid}, \text{ssid}, C, \text{flag}, \text{pw}^*)$  from  $\mathcal{A}^*$ , if  $\exists \text{rec} = (\text{sid}, \text{ssid}, U', \mathbf{P}_{\text{sid}}, \text{pw}, \text{pw}', \cdot, \cdot)$  not marked as  $\text{pretested}(c)$  for any  $c$  then:
  - [S.P.1] if  $\text{flag} = 1$ ,  $|C| = t+1$ , and  $\forall S \in C (\text{tx}(S) > 0)$ , then set  $\text{tx}(S)--$  for all  $S \in C$ , set  $b := (\text{pw}' == \text{pw})$ , send  $b$  to  $\mathcal{A}^*$  and mark  $\text{rec}$  as  $\text{pretested}(b)$ ;
    - [S.P.1\*] moreover, if  $b = 1$  and  $U' \in \text{Corr} \cup \{\mathcal{A}^*\}$  set  $\text{flag}_{\text{sid}} = 1$ ;
  - [S.P.2] if  $\text{flag} = 2$  then set  $b := (\text{pw}' == \text{pw}^*)$ , send  $b$  to  $\mathcal{A}^*$ , and if  $b = 1$  then mark  $\text{rec}$  as  $\text{pretested}(2)$  else mark  $\text{rec}$  as  $\text{pretested}(0)$ ;
- [S.F] On  $(\text{ptsig.finsign}, \text{sid}, \text{ssid}, C', \text{flag}, \sigma^*, m^*)$  from  $\mathcal{A}^*$ , retrieve  $\text{rec} = (\text{sid}, \text{ssid}, U', \mathbf{P}_{\text{sid}}, \text{pw}, \text{pw}', V, m)$  and do:
  - [S.F.0] if  $m = \perp$  and  $U' \in \text{Corr} \cup \{\mathcal{A}^*\}$  reset  $m := m^*$ ;
  - [S.F.1] if  $\text{flag}_{\text{sid}} = 1$ ,  $\text{rec}$  is marked  $\text{pretested}(0)$ , and  $U' \in \text{Corr} \cup \{\mathcal{A}^*\}$ , then change  $\text{rec}$  mark to  $\text{pretested}(1)$ ;
  - [S.F.F] send  $(\text{ptsig.finsign}, \text{sid}, \text{ssid}, m, \sigma)$  to  $U'$  s.t.
    - [S.F.F.1] if  $\text{flag} = 1$ ,  $\text{rec}$  is marked  $\text{pretested}(1)$ ,  $|C'| = t+1$ ,  $C' \subseteq \mathbf{P}_{\text{sid}}$ ,  $\exists$  record  $(\text{sid}, m, S)$  for all  $S \in C'$ ,  $V(m, \sigma^*) = 1$ , and there is no saved record  $(\text{sid}, m, \sigma^*, 0)$ , then save record  $(\text{sid}, m, \sigma^*, 1)$  and set  $\sigma := \sigma^*$ ;
    - [S.F.F.2] if  $\text{flag} = 2$  and  $\text{rec}$  is marked  $\text{pretested}(2)$  then set  $\sigma := \sigma^*$ ;
    - [S.F.F.3] if neither of the above two cases is met set  $\sigma := \perp$ .

**Verification:**

- On  $(\text{ptsig.verify}, \text{sid}, m, \sigma, V)$  from  $Q$ , send  $(\text{ptsig.verify}, \text{sid}, m, \sigma, V)$  to  $\mathcal{A}^*$  and do:
  - [V.1] if  $\exists$  records  $(\text{sid}, \mathbf{P}_{\text{sid}}, \text{pw}, V)$  and  $(\text{sid}, m, \sigma, \beta')$  then set  $\beta := \beta'$ ;
  - [V.2] else, if  $\exists$  record  $(\text{sid}, \mathbf{P}_{\text{sid}}, \text{pw}, V)$  but no  $(\text{sid}, m, \sigma, 1)$  for any  $\sigma$  then set  $\beta := 0$ ;
  - [V.3] else set  $\beta := V(m, \sigma)$ .
- [V.V] Record  $(\text{sid}, m, \sigma, \beta)$  and send  $(\text{ptsig.verified}, \text{sid}, m, \sigma, \beta)$  to  $Q$ .

**Fig. 5.**  $\mathcal{F}_{\text{aptSIG}}$ : Ideal Functionality for Password-Protected Threshold Signature



**Fig. 6.** Adversarial Interfaces of  $\mathcal{F}_{\text{aptSIG}}$

$(\text{ptsig.uit}, \text{sid}, \text{V})$  from  $\mathcal{A}^*$  models the ideal-world adversary allowing an initialization instance identified by  $\text{sid}$  to complete, and  $\text{U}$  to output the public key  $\text{V}$ . Note that all parties input the identities of all participants into the protocol, and  $\mathcal{F}_{\text{aptSIG}}$  reacts to query  $\text{ptsig.uit}$  only if all intended parties participate in the initialization. This is realizable if  $\text{U}$  and each  $\text{S}_i$  can authenticate each other, and our  $\text{aptSIG}$  protocol indeed relies on authenticated channels in initialization. The public key  $\text{V}$  is associated with initialization identifier  $\text{sid}$  in the sense that  $\text{sid}$  serves as a handle to the *password-protected secret-sharing* ( $\text{ppss}$ ) of a private signing key corresponding to  $\text{V}$ . (Functionality  $\mathcal{F}_{\text{aptSIG}}$  does not ensure that this sharing is successfully established when  $\text{U}$  outputs  $\text{V}$ , but  $\mathcal{F}_{\text{aptSIG}}$  allows  $\text{U}$  to verify it, e.g. if  $\text{U}$  invokes the signing protocol on a test message.)

Once key  $\text{V}$  is created, query  $(\text{ptsig.usign}, \text{sid}, \text{ssid}, \text{S}, \text{pw}', \text{m})$  from  $\text{U}'$  models user  $\text{U}'$  (possibly using a different platform than  $\text{U}$ , hence a different name tag  $\text{U}'$ ) who holds password  $\text{pw}'$  (which might or might not equal to  $\text{pw}$ ) starting a signing protocol instance on message  $\text{m}$  and a  $\text{ppss}$ -protected key identified by  $\text{sid}$ . Identifier  $\text{ssid}$  is a handle of  $\text{U}'$  on that instance, and  $\text{S}$  is a subset of  $t + 1$  servers with whom  $\text{U}$  intends to communicate. However,  $\mathcal{F}_{\text{aptSIG}}$  doesn't enforce authentication in signing, and the signing instance record it creates,  $(\text{sid}, \text{ssid}, \text{U}', \mathbf{P}_{\text{sid}}, \text{pw}, \text{pw}', \text{V}, \text{m})$  ignores field  $\text{S}$ . Query  $(\text{ptsig.ssign}, \text{sid}, \text{ssid}, \text{U}', \text{m})$  from  $\text{S}$  models  $\text{S}$  agreeing to sign  $\text{m}$  using the  $\text{ppss}$ -protected key identified by  $\text{sid}$ . Field  $\text{U}'$  is a counterparty address,  $\text{ssid}$  is  $\text{S}'$ 's local instance handle, but they play no security roles and  $\mathcal{F}_{\text{aptSIG}}$  ignores them. In particular,  $\mathcal{F}_{\text{aptSIG}}$  does not enforce equality of  $\text{ssid}$  or  $\text{U}'$  tags used by the participants in signing.

**(2)  $\mathcal{F}_{\text{aptSIG}}$ : Signature Completion.** Signing protocol output is controlled by two queries by an ideal-world adversary  $\mathcal{A}^*$ :  $\text{ptsig.pretest}$  and  $\text{ptsig.finsign}$ .  $\mathcal{F}_{\text{aptSIG}}$  associates servers  $\text{S} \in \mathbf{P}_{\text{sid}}$  with ticket counters  $\text{tx}(\text{S})$ , as in the  $\text{aPPSS}$  functionality  $\mathcal{F}_{\text{aPPSS}}$  of Sect. 3, and each  $\text{S}$  can trigger  $\mathcal{F}_{\text{aptSIG}}$  to record  $(\text{sid}, \text{m}, \text{S})$  which stands for  $\text{S}$  agreeing to sign  $\text{m}$ , as in the  $\text{tSIG}$  functionality  $\mathcal{F}_{\text{tSIG}}$  of Sect. 2. When  $\text{S}$  issues a query  $(\text{ptsig.ssign}, \dots, \text{m})$  then  $\mathcal{F}_{\text{aptSIG}}$  increments  $\text{tx}(\text{S})$  and records  $(\text{sid}, \text{m}, \text{S})$  at the same time.

Queries `ptsig.pretest` and `ptsig.finsign` serve two purposes: The first one, denoted by  $\mathcal{A}^*$  using `flag=1`, is a passive completion of the signing instance. First,  $\mathcal{A}^*$  can use `ptsig.pretest` with `flag=1` to “pre-complete” that instance and learn if party  $U'$  runs the protocol on the correct password  $\text{pw}' = \text{pw}$ . This is akin to `TestAbort` query in the UC aPAKE model [26]: A protocol can make it detectable whether  $U'$  runs on the correct password, e.g. because otherwise  $U'$  aborts, in which case the adversary learns if  $\text{pw}' = \text{pw}$  by observing the protocol. In this test,  $\mathcal{A}^*$  must specify a subset  $\mathbf{C}$  of  $t + 1$  servers with non-zero ticket counters (which  $\mathcal{F}_{\text{aptSIG}}$  decrements), which enforces that  $U'$  finalization requires  $t + 1$  participating servers. Note that these servers can run on different messages than  $U'$ , i.e.  $\mathcal{A}^*$  can mix and match  $S$  sessions in completing `ptsig.pretest`.

If  $\text{pw}' = \text{pw}$  then  $\mathcal{A}^*$  can follow up the `(ptsig.pretest, ..., flag=1, ...)` query with `(ptsig.finsign, ..., C', flag=1,  $\sigma^*$ ,  $\perp$ )`, which corresponds to finalizing the signing instance on message  $m$  with signature  $\sigma^*$ . Indeed, if  $U'$  runs on the correct password and the attacker is passive then  $U'$  can output a signature.  $\mathcal{F}_{\text{aptSIG}}$  processes this query in the same way as the threshold signature functionality  $\mathcal{F}_{\text{tSIG}}$  of Sect. 2, i.e. it checks that  $t + 1$  servers in subset  $\mathbf{C}'$  agreed to sign  $m$ , that  $\sigma^*$  was not previously recorded as a faulty signature, and that  $V(m, \sigma^*) = 1$ , and if all conditions are met then it outputs  $\sigma^*$  to  $U'$  and declares  $\sigma^*$  as a valid signature on  $m$  by recording a “signature” tuple  $(sid, m, \sigma^*, 1)$ . These tuples control the outputs of a signature verification query `ptsig.verify`, and  $\mathcal{F}_{\text{aptSIG}}$  handles that exactly as  $\mathcal{F}_{\text{tSIG}}$ , i.e. if there is no recorded tuple  $(sid, m, \sigma^*, 1)$  then `(ptsig.verify, ..., m,  $\sigma^*$ , V)` query should return 0.

We note that  $\mathcal{F}_{\text{aptSIG}}$  does not enforce that  $\mathbf{C}' = \mathbf{C}$ , i.e. the adversary is allowed to mix-and-match servers and use a different subset  $\mathbf{C}$  of server instances to “pre-complete” a signature session via the `ptsig.pretest` query, and a different subset  $\mathbf{C}'$  to complete the session via the `ptsig.finsign` query. Moreover, the second set of servers must be signing  $m$ , but the first one might not. We allow this “disconnection” in  $\mathcal{F}_{\text{aptSIG}}$  to enable an efficient aptSIG protocol of Sect. 4.2, which does not enforce  $\mathbf{C}' = \mathbf{C}$ . However, the practical import of adversary replacing part of  $m$ -signing server session with parts taken from some  $m'$ -signing server session seems innocuous, given that in the end a signature on  $m$  cannot be created unless a  $\text{pw}$ -holding user and  $t + 1$  servers all agree to it.

**(3)  $\mathcal{F}_{\text{aptSIG}}$ : Active Attacks.** The first type of active attack is an on-line password guessing attack against honest servers, where  $\mathcal{A}^*$  poses as a user, or employs a corrupt user  $U'$ , and runs a signing protocol via interface `ptsig.usign` on some password  $\text{pw}'$  ([S.U]), followed by `ptsig.pretest` and `ptsig.finsign` with `flag=1` ([S.P.1]). The same logic as above will apply to this sequence, except since the adversary contributed  $\text{pw}'$  in `ptsig.usign`, the same interface will reveal if  $\text{pw}' = \text{pw}$  (in [S.P.1] the functionality sends this bit to the adversary). Moreover, each ptSIG instance  $sid$  is associated with a flag  $\text{flag}_{sid}$  which switches from 0 to 1 if it ever happens that the adversary found password  $\text{pw}'$  in this way ([S.P.1\*]) (or via offline attacks, see below). The consequence of  $\text{flag}_{sid} = 1$  is that any adversarial signing instance, even one that starts with an incorrect password  $\text{pw}'$ , and consequently its reconstruction record `rec` would be marked `pretested(0)` in

`ptsig.pretest`, is effectively treated in `ptsig.finsign` as if it was marked `pretested(1)`, which means that the functionality will “sign” message  $m^*$  in this signing session (as long as  $t + 1$  servers also agree to sign it) ([S.F.1]). In other words, if the adversary guesses the right password on some ptSIG session, then we allow him to “late switch” any incorrect password to the correct one on all his other signing sessions.

The second type of active attack is an on-line password guessing attack against an honest user. This is modeled via `ptsig.pretest` ([S.P.2]) and `ptsig.finsign` queries with `flag = 2` ([S.F.F.2]). Here  $\mathcal{A}^*$  can set  $C = \perp$ , but must enter a password guess  $pw^*$ , and in `ptsig.pretest` it will learn if  $pw' = pw^*$  where  $pw'$  is a password used by an honest user  $U'$  ([S.P.2]). If not then  $U'$  can subsequently only abort, but if so then subsequent `ptsig.finsign` makes  $U'$  output as signature an arbitrary value  $\sigma^*$  chosen by  $\mathcal{A}^*$  ([S.F.F.2]). This reflects the fact that the only security hedge which  $U'$  enters into signing is its password  $pw'$ , so if an online attacker guesses  $pw'$ , the attacker can wlog. run aptSIG initialization on  $pw'$  and then run the aptSIG signing on the resulting values, thus making  $U'$  output e.g. a signature on  $m$  but issued by an adversarial key. However, this attack does not imply signature forgery, because  $\mathcal{F}_{\text{aptSIG}}$  does not add tuple  $(sid, m, \sigma^*, 1)$  to its records. In particular, a user could run signature verification (`ptsig.verify, sid, m, \sigma^*, V`) on its aptSIG output, and in case of the above attack she would learn that  $\sigma^*$  is not a valid signature **and** that she was subject of an active attack by someone who learned her password  $pw'$ .<sup>3</sup>

**(4)  $\mathcal{F}_{\text{aptSIG}}$ : Adaptive Server Corruptions and ODA.** Adversary  $\mathcal{A}^*$  can adaptively corrupt any server  $S$  ([SC]), which allows  $\mathcal{A}^*$  to (1) freely issue tickets for  $S$ , using `ptsig.ssign` with  $b = 1$ , and (2) freely issue  $S$ 's “partial signatures” on arbitrary messages  $m$ , using `ptsig.ssign` with  $m \neq \perp$  ([S.S]). The latter actions can result in signatures if  $U$  using the correct password  $pw' = pw$  wants to sign the same  $m$  ([S.F.F.1]), or if the attacker learns  $pw$  and invokes user-side on that  $pw$  and  $m$ . The former actions allow the attacker to test passwords via command `ptsig.testpw`, which lets  $\mathcal{A}^*$  exchange  $t + 1$  tickets from some  $t + 1$  servers for an off-line test of *one* password guess  $pw^*$  specified by  $\mathcal{A}^*$ . Note that corrupt  $S_i$ 's these tickets are “free” to  $\mathcal{A}^*$  so after corrupting  $t + 1$  servers these tests can be done fully offline, but if  $\mathcal{A}^*$  needs to add the tickets from honest servers to this mix then only one such ticket is created in each signing instance  $S_i$  runs, i.e. if adversary corrupts  $t' \leq t + 1$  servers then it can test  $q$  passwords only by on-line interactions with  $q * (t + 1 - t')$  servers ([PT]).

Crucially, *even if all servers are corrupted*, attacker  $\mathcal{A}^*$  has no avenue to forge message signatures unless  $\mathcal{A}^*$  finds out user's password  $pw$  and runs `ptsig.usign` (e.g. as corrupt  $U'$ ) on  $pw$ . (Moreover, if fewer than  $t + 1$  servers are corrupt than even knowing  $pw$  lets  $\mathcal{A}^*$  sign only messages which some uncorrupted servers agree to sign.) Moreover, the only avenues to finding password  $pw$  ([PT]) consist of (1) *online* guessing attacks against either the servers or the user as long as

<sup>3</sup>  $\mathcal{F}_{\text{aptSIG}}$  lets  $\mathcal{A}^*$  set the user instance's message  $m$  to arbitrary  $m^*$  in the finalization of the signing protocol, but only for adversarial user instances, i.e. we allow adversarial signing instances to “late-commit” to their messages.

$\mathcal{A}^*$  corrupts fewer than  $t + 1$  servers, and (2) (fully) *offline* dictionary attacks (ODA), as explained above, enabled once  $\mathcal{A}^*$  corrupts  $t + 1$  servers.

**User/Message Authentication and Perfect Forward Secrecy.** In the aptSIG ideal model  $\mathcal{F}_{\text{aptSIG}}$ , when servers sign they take input  $m$  from the environment, and they do not know if their counterparty holds the right password, and even if they do then whether they authorize signing this message. A model which assures both properties extends the aptSIG model to capture perfect forward security (PFS), because it would imply that if no password-holding entity wants to sign some message at a given time, then the adversary who might capture the password in the future, cannot “redo” these signature instances, and can only use the compromised password on new signature sessions.

The PFS property can be added in black-box way by running two instances of aptSIG: Consider a modified signing protocol which executes two instances of aptSIG, first one on the message  $m$  concatenated with nonce *ssid*, and only if this one creates a valid signature on the  $m, \text{ssid}$ , then the proper aptSIG instance would execute on just  $m$ . The first aptSIG instance accomplishes the above requirements, because only a correct password could have caused this aptSIG instance to issue a valid signature on the  $m, \text{ssid}$  pair.

In the full version of the paper we define a PFS version of the aptSIG ideal model, denoted  $\mathcal{F}_{\text{aptSIG-PFS}}$ , and we show that the efficient aptSIG scheme which we show in the next subsection, can be adapted more efficiently to implement the PFS property. The idea is very similar to the one above except that the first instance of aptSIG is replaced by a standard signature made on pair  $m, \text{ssid}$  by the user  $U$ . Indeed, efficiency-wise the PFS protocol variant shown in the full version of the paper adds only the cost of issuing a single standard signature for user  $U$  and a signature verification for each server  $S$ .

## 4.2 Generic AptSIG Protocol

In Fig. 7 we show a generic construction of an augmented password-protected threshold signature (aptSIG), using an augmented Password-Protected Secret Sharing (aPPSS) and a Threshold Signature (tSIG). The protocol in addition relies on functionality  $\mathcal{F}_{\text{AUTH}}$  but it is used only in initialization. The protocol also relies on an Equivocable Authenticated Encryption scheme, denoted AE.

**Threshold Signature Protocol  $\Pi_{\text{tSIG}}$ .** In the description of protocol  $\Pi_{\text{tSIG}}$  in Fig. 7, we don’t use the threshold signature functionality  $\mathcal{F}_{\text{tSIG}}$ , but use the tSIG protocol directly. We choose this way of describing the aptSIG scheme because whereas the server parties  $P_i \in \mathbf{P}$  can store secret state between tSIG initialization and signature phases, the user party  $U$  is assumed to have no secure storage (except for memorizing the password), hence it is in particular incapable of locally storing the secret share generated in key generation of tSIG. Indeed, we use the aPPSS scheme together with the authenticated encryption AE to “securely transmit” this user’s tSIG state between initialization and signature phase, but since this secure transmission can fail, i.e., in case of successful

Public parameters: Security parameter  $\lambda$ , threshold parameters  $t, n$  s.t.  $t \leq n$ .  
 Let  $\text{AE} = (\text{AuthEnc}, \text{AuthDec})$  be an Equivocable Authenticated Encryption, and let  $\text{tSIG} = (\Pi_{\text{TKeyGen}}, \Pi_{\text{TSign}}, \Pi_{\text{TVerify}})$  be a Threshold Signature scheme realizing functionality  $\mathcal{F}_{\text{tSIG}}$  (see text).  $\text{add}(sid, U)$  parses  $sid = (sid', \mathbf{P}_{sid})$  and outputs  $sid^+ = (sid', \mathbf{P}_{sid}^+)$  s.t. if  $\mathbf{P}_{sid} = (P_1, \dots, P_n)$  then  $\mathbf{P}_{sid}^+ = (U, P_1, \dots, P_n)$ .

**Initialization for user U:**

1. On input  $(\text{ptsig.uit}, sid, pw)$ , send  $(\text{ppss.uit}, sid, pw, \perp)$  to  $\mathcal{F}_{\text{aPPSS}}$ , and let  $sk$  denote  $\mathcal{F}_{\text{aPPSS}}$ 's output.
2. Run  $\text{tSIG}.\Pi_{\text{TKeyGen}^+}$  on input  $sid^+ = \text{add}(sid, U)$ . Let  $(ts_U, tcs_U)$  and  $V$  be resp. U's local output and the generated public key.
3. Set  $\text{aec}_U := \text{AE.AuthEnc}_{sk}(U, ts_U, tcs_U)$ , send  $(\text{send}, sid, P_i, \text{aec}_U)$  to  $\mathcal{F}_{\text{AUTH}}$  for all  $P_i \in \mathbf{P}_{sid}$ , output  $(\text{ptsig.verificationkey}, sid, V)$ .

**Initialization for server S:**

1. On input  $(\text{ptsig.sinit}, sid, i, U)$  send  $(\text{ppss.sinit}, sid, i, U)$  to  $\mathcal{F}_{\text{aPPSS}}$  and run  $\text{tSIG}.\Pi_{\text{TKeyGen}^+}$  on  $sid^+ = \text{add}(sid, U)$ . Let  $(ts_i, tcs_i)$  be S's local output.
2. On message  $(\text{sent}, sid, U, S, \text{aec}_U)$  from  $\mathcal{F}_{\text{AUTH}}$ , save  $(sid, sid^+, ts_i, tcs_i, \text{aec}_U)$ .

**Signing for user U'**

1. On input  $(\text{ptsig.usign}, sid, ssid, \mathbf{S}, pw', m)$  for  $|\mathbf{S}| \geq t+1$  from  $U'$ , send  $(\text{ppss.urec}, sid, ssid, \mathbf{S}, pw')$  to  $\mathcal{F}_{\text{aPPSS}}$ , and wait to receive  $(\text{ppss.urec}, sid, ssid, sk)$  from  $\mathcal{F}_{\text{aPPSS}}$  and message  $(sid, \text{aec}_U)$  from all  $S \in \mathbf{S}$ .
2. Output  $(\text{ptsig.usign}, sid, ssid, m, \perp)$  and abort if either (1)  $sk = \perp$ , or (2) it is not the case that all  $S \in \mathbf{S}$  send the same message  $(sid, \text{aec}_U)$ , or (3)  $\text{AE.AuthDec}_{sk}(\text{aec}_U)$  returns  $\perp$ .
3. Otherwise, let  $(U, ts_U, tcs_U) = \text{AE.AuthDec}_{sk}(\text{aec}_U)$ , set  $sid^+ = \text{add}(sid, U)$ , run protocol  $\text{tSIG}.\Pi_{\text{TSign}^+}$  on input  $(sid^+, ts_U, tcs_U, m)$ , and when this protocol outputs  $\sigma$ , output  $(\text{ptsig.finsign}, sid, ssid, m, \sigma)$ .

**Signing for server S**

1. On input  $(\text{ptsig.ssign}, sid, ssid, U', m)$  from  $S$ , retrieve stored tuple  $(sid, sid^+, ts_i, tcs_i, \text{aec}_U)$ , send  $(\text{ppss.srec}, sid, ssid, U')$  to  $\mathcal{F}_{\text{aPPSS}}$ , send  $(sid, \text{aec}_U)$  to  $U'$ , and run  $\text{tSIG}.\Pi_{\text{TSign}^+}$  on input  $(sid^+, ts_i, tcs_i, m)$ .

**Verification for Q**

1. On input  $(\text{ptsig.verify}, sid, m, \sigma, V)$  from  $Q$ , runs  $\beta = \text{tSIG}.\Pi_{\text{TVerify}}(V, m, \sigma)$ , and output  $(\text{ptsig.verified}, sid, m, \sigma, \beta)$

**Fig. 7.** Protocol  $\Pi_{\text{aptSIG}}$  which realizes  $\mathcal{F}_{\text{aptSIG}}$  in  $(\mathcal{F}_{\text{aPPSS}}, \mathcal{F}_{\text{AUTH}})$ -hybrid world

password-guessing attack on aPPSS, an honest user may execute tSIG on adversarially chosen inputs. In essence, our aptSIG protocol runs the real-world tSIG protocol rather than an ideal functionality  $\mathcal{F}_{\text{tSIG}}$ , because functionality  $\mathcal{F}_{\text{tSIG}}$  does not support a party running the signing protocol on the inputs which do not correspond to the state created by the key generation for this party. Note that this proof technique was used in the analysis of the OPAQUE protocol [32], for the same reason that a UC-secure protocol tool, UC AKE in OPAQUE and UC tSIG here, is used within a protocol on keys which might not match the ones prescribed by the protocol.

**tSIG Functionality and Communication Setting.** We assume that the tSIG scheme consists of (1) protocol  $\Pi_{\text{TKeyGen}}$ , which implements  $\mathcal{F}_{\text{tSIG}}$  command (tsig.keygen,  $sid'$ ) for  $sid' = (sid, \mathbf{P}^+)$ ; (2) protocol  $\Pi_{\text{TSign}}$ , which implements  $\mathcal{F}_{\text{tSIG}}$  command (tsig.sign,  $sid, m$ ); and (3) algorithm  $\Pi_{\text{TVerify}}(\mathbf{V}, m, \sigma)$  which implements (tsig.verify,  $sid, m, \sigma, \mathbf{V}$ ), which simply returns  $\mathbf{V}(m, \sigma)$ . Note that set  $\mathbf{P}^+$  is a list of  $n + 1$  tSIG participants, and we form it by prepending the user party identifier  $\mathbf{U}$  to the list of server identifiers  $\mathbf{P} = \{\mathbf{P}_1, \dots, \mathbf{P}_n\}$ .

We use  $\text{ts}_i$  to denote the state created for player  $\mathbf{P}_i$  by the distributed key generation protocol  $\Pi_{\text{TKeyGen}}$ , including  $i = \mathbf{U}$ . (In the following we will use  $\mathbf{P}_{\mathbf{U}}$  and  $\mathbf{U}$  interchangeably.) However, many threshold signature schemes assume that protocol parties have access to some additional secure communication channels, in the very least secure point-to-point channels and often also a reliable authenticated broadcast channel. (These are the communication assumptions of most work on threshold cryptosystems, including e.g. the UC threshold ECDSA of [13] and the threshold BLS scheme in Sect. 2, albeit the latter only in the initialization phase.) Whereas aptSIG servers can be connected by such channels, we cannot assume this for the user. Indeed, in aptSIG initialization we assume user  $\mathbf{U}$  has only point-to-point authenticated channels with each server  $\mathbf{S}_i$ , and in aptSIG signing we assume a plain network. If threshold signature protocols  $\Pi_{\text{TKeyGen}}$  and/or  $\Pi_{\text{TSign}}$  make such communication assumptions, in the initialization phase our aptSIG prepends protocol  $\Pi_{\text{TKeyGen}}$  with a subprotocol which adds  $\mathbf{U}$  to this assumed communication setting.

For the above communication setting, this subprotocol could work as follows: Since in aptSIG initialization  $\mathbf{U}$  and each  $\mathbf{S}_i$  have pairwise authenticated channels, these can be upgraded to secure channels using key exchange, e.g. Diffie-Hellman, executed between  $\mathbf{U}$  and each  $\mathbf{S}_i$ . As for authenticated broadcast, it is typically implemented using PKI (e.g. assuming partial synchrony and reliable message delivery between uncorrupted parties), in which case  $\mathbf{U}$  can generate a signing key, deliver it over authenticated channels to each  $\mathbf{S}_i$ , and  $\mathbf{S}_i$ 's can agree on it using their authenticated broadcast channels. Likewise, each  $\mathbf{S}_i$  can send the list of all servers' public keys to  $\mathbf{U}$ , and  $\mathbf{U}$  can reject unless all the lists are the same. We denote this extended  $\Pi_{\text{TKeyGen}}$  protocol as  $\Pi_{\text{TKeyGen}^+}$ , and we use  $\text{tcs}_i$  to denote the secure communication tokens each  $\mathbf{P}_i$  retains from it in subsequent  $\Pi_{\text{TKeyGen}}$  and  $\Pi_{\text{TSign}}$  executions. Whereas each server  $\mathbf{S}_i$  can update its pre-existing communication tokens with  $\text{tcs}_i$ 's output by  $\Pi_{\text{TKeyGen}^+}$ , user  $\mathbf{U}$  cannot retain state between executions. However, we solve this by adding

the communication tokens  $\text{tcs}_U$  to the threshold signature state  $\text{ts}_U$  created by  $\Pi_{\text{TKeyGen}}$ , and we encrypt both using the aPPSS-protected key.

**Equivocable Authenticated Encryption.** Protocol  $\Pi_{\text{aptSIG}}$  uses symmetric Authenticated Encryption scheme  $\text{AE} = (\text{AuthEnc}, \text{AuthDec})$  to encrypt the local state of  $U$  output by  $\Pi_{\text{TKeyGen}^+}$ . We denote this state as  $(U, \text{ts}_U, \text{tcs}_U)$ , where  $\text{ts}_U, \text{tcs}_U$  are explained above, and identity  $U$  needs to be retained as well because tSIG assumes that its identifier  $\text{sid}^+$  includes the identities of all tSIG participants, i.e.  $\mathbf{P}^+ = (U, P_1, \dots, P_n)$ , and aptSIG should allow the user to retrieve signatures using the password only, i.e. it should not rely on the user remembering the identifier  $U$  used in the initialization.

We need the authenticated encryption to have *ciphertext integrity* under a single chosen message attack. This is a relaxation of standard ciphertext integrity security notion for authenticated encryption. We also require the scheme  $\text{AE}$  to be *equivocable*, i.e. in the scenario where the adversary gets a ciphertext followed by the key, there is a simulator that can create an indistinguishable ciphertext with no information about the plaintext except for its length, and then create the key to decrypt this ciphertext to any given plaintext. Formally, we call an (authenticated) encryption  $\text{AE}$  *equivocable* if there is an efficient simulator  $\text{SIM}$  s.t. for any efficient algorithm  $\mathcal{A}$ , the distinguishing advantage  $\text{Adv}_{\mathcal{A}}^{\text{EQV}, \text{AE}}(\lambda) = |p_{\mathcal{A}}^0 - p_{\mathcal{A}}^1|$  is a negligible function of  $\lambda$ , where  $p_{\mathcal{A}}^b = \Pr[1 \leftarrow \mathcal{A}(\text{st}_{\mathcal{A}}, \text{aec}, \text{sk}) \mid (\text{st}_{\mathcal{A}}, \text{aec}, \text{sk}) \leftarrow \text{Exp}^b]$ , and

$$\begin{aligned} \text{Exp}^0 &: (m, \text{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(\lambda), \text{sk} \leftarrow \{0, 1\}^\lambda, \text{aec} \leftarrow \text{AuthEnc}_{\text{sk}}(m) \\ \text{Exp}^1 &: (m, \text{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(\lambda), (\text{aec}, \text{st}_{\mathcal{S}}) \leftarrow \text{SIM}(|m|), \text{sk} \leftarrow \text{SIM}(\text{st}_{\mathcal{S}}, m) \end{aligned}$$

Note that equivocability implies standard semantic security of encryption. In the following we will use the term *Equivocable Authenticated Encryption* if encryption is (1) equivocable and (2) has ciphertext integrity. These properties are easy to achieve in an idealized model like ROM [32], e.g.  $E(\text{sk}, m) = m \oplus G(\text{sk})$  is equivocable if  $G$  is a random oracle. If an equivocable encryption is extended to authenticated encryption, e.g. by computing a MAC on the ciphertext, this achieves ciphertext integrity but does not effect equivocation because the authentication mechanism is computed over the ciphertext.

### 4.3 Security of the AptSIG Protocol

**Simulation Overview.** We construct a simulator  $\text{SIM}$  which will show that no environment  $\mathcal{Z}$  can distinguish the ideal-world and real-world interactions. Since protocol  $\Pi_{\text{aptSIG}}$  relies on the UC security of three components, aPPSS, tSIG, and AUTH, we first overview how the real world and the ideal world interactions involve the protocols, functionalities, or simulators of these components.<sup>4</sup> Without loss of generality we assume a “dummy” adversary  $\mathcal{A}^*$  that is an adversary

<sup>4</sup> Due to space constraints we defer to the full version of the paper, which captures the top-level view of these interactions in the real-world and ideal-world executions.



who merely passes all its messages and computations to the environment  $\mathcal{Z}$ . Our proof assumes that the real execution happens in the  $(\mathcal{F}_{\text{aPPSS}}, \mathcal{F}_{\text{AUTH}})$ -hybrid world, and below we omit the details of interactions with the adversary where in the ideal world  $\text{SIM}$  will emulate  $\mathcal{F}_{\text{aPPSS}}$  and  $\mathcal{F}_{\text{AUTH}}$ , because that part of the simulation is trivial:  $\text{SIM}$  gains all the information needed from  $\mathcal{A}^*$ 's interfaces to these functionalities, and simply follows the code of  $\mathcal{F}_{\text{aPPSS}}$  and  $\mathcal{F}_{\text{AUTH}}$ .

Simulator  $\text{SIM}$  interacts with the ideal functionality  $\mathcal{F}_{\text{aptSIG}}$ , which in turn interacts with the environment  $\mathcal{Z}$  via “dummy” honest parties playing the role of either user  $U$  and server(s)  $S$ . The environment  $\mathcal{Z}$  can also instruct  $\mathcal{A}^*$  to send malicious inputs to  $\text{SIM}$  on behalf of corrupt or compromised parties, e.g. compromised servers. There are three types of  $\text{SIM}$ - $\mathcal{A}^*$  interactions, corresponding to three difference interfaces  $\mathcal{A}^*$  has in the real world. First, there is the *network* interface, i.e. messages which protocol  $\Pi_{\text{aptSIG}}$  sends over plain channels. This interface is used solely for sending  $\text{aec}_U$  in the signing protocol. Second,  $\mathcal{A}^*$  can communicate with functionalities  $\mathcal{F}_{\text{AUTH}}$  and  $\mathcal{F}_{\text{aPPSS}}$ , which  $\text{SIM}$  will emulate in the ideal-world. Third, since protocol  $\Pi_{\text{aptSIG}}$  runs the real-world protocol  $\Pi_{\text{tSIG}}$  instead of using  $\mathcal{F}_{\text{tSIG}}$  as a black-box (see also the explanation above),  $\mathcal{A}^*$  expects to interact with  $\Pi_{\text{tSIG}}$  instances. In the ideal-world,  $\text{SIM}$  will not execute the real-world protocol  $\Pi_{\text{tSIG}}$ , and instead it will delegate this to a simulator  $\text{SIM}_{\text{tSIG}}$  (the simulator whose existence is implied by the assumption that protocol  $\Pi_{\text{tSIG}}$  UC-realizes functionality  $\mathcal{F}_{\text{tSIG}}$ ), which  $\text{SIM}$  will execute as a black-box. Simulator  $\text{SIM}_{\text{tSIG}}$  can emulate execution of  $\Pi_{\text{tSIG}}$  instances to  $\mathcal{A}^*$  if  $\text{SIM}_{\text{tSIG}}$  interacts with the ideal functionality  $\mathcal{F}_{\text{tSIG}}$ . Therefore,  $\text{SIM}$  will implement an “ $\mathcal{F}_{\text{tSIG}}$ ” interface (just like the “ $\mathcal{F}_{\text{AUTH}}$ ” and “ $\mathcal{F}_{\text{aPPSS}}$ ” interfaces described above) on which it will talk not to  $\mathcal{A}^*$  but to  $\text{SIM}_{\text{tSIG}}$ . Note that from  $\text{SIM}$ 's perspective  $\text{SIM}_{\text{tSIG}}$  can be thought of as an extension of adversary  $\mathcal{A}^*$  (indeed  $\text{SIM}$  treats  $\text{SIM}_{\text{tSIG}}$  as a black box, just like it treats  $\mathcal{A}^*$ ), at which point  $\text{SIM}$ 's goals is just to correctly emulate the “ $\mathcal{F}_{\text{tSIG}}$ ” interface with  $\text{SIM}_{\text{tSIG}}$ .

As discussed above, there is one further unusual aspect of the simulation: In one special case, which corresponds to an honest party  $U$  recovering wrong  $\text{tSIG}$  shares because of a successful online active attack against  $U$ 's password in the  $\text{aPPSS}$  subprotocol, the real-world execution in this case involves  $U$  running the  $\text{tSIG}$  signing subprotocol on *adversarial inputs* rather than the inputs prescribed for  $U$  in the  $\text{tSIG}$  key generation. Such honest party's execution is not supported by functionality  $\mathcal{F}_{\text{tSIG}}$ , so the simulator cannot send any messages on the “ $\mathcal{F}_{\text{tSIG}}$ ” interface to  $\text{SIM}_{\text{tSIG}}$  to emulate such  $\text{tSIG}$  signing protocol instances on behalf of  $U$ . Instead,  $\text{SIM}$  will simply execute itself the  $\text{tSIG}$  instance on behalf of  $U$  on such adversarial inputs. (Note that  $\text{SIM}$  learns from the “ $\mathcal{F}_{\text{aPPSS}}$ ” interface the adversarial inputs which the real-world  $U$  would use, because the adversary sends them to the real-world  $U$  via functionality  $\mathcal{F}_{\text{aPPSS}}$ ) This  $U$  instance can be thought of as another extension of the adversary, and  $\text{SIM}$  will inform  $\text{SIM}_{\text{tSIG}}$  (and pass to  $\mathcal{A}^*$ ) whatever this instance sends e.g. to honest  $\text{tSIG}$  servers, which are emulated by  $\text{SIM}_{\text{tSIG}}$ .

**Theorem 3.** *If  $\text{AE} = (\text{AuthEnc}, \text{AuthDec})$  is an Equivocable Authenticated Encryption, and  $\text{tSIG} = (\Pi_{\text{TKeyGen}}, \Pi_{\text{TSign}}, \Pi_{\text{TVerify}})$  is a Threshold Signature*

scheme which UC-realizes functionality  $\mathcal{F}_{\text{tSIG}}$ , then protocol  $\Pi_{\text{aptSIG}}$  in Fig. 7 UC-realizes functionality  $\mathcal{F}_{\text{aptSIG}}$  in Fig. 5 in the  $(\mathcal{F}_{\text{aPPSS}}, \mathcal{F}_{\text{AUTH}})$ -hybrid model.

Due to space constraints, we defer the detailed specification of the simulator SIM, as well as the rest of the proof of Theorem 3, to the full version of the paper [21].

## 5 Concrete Instantiation of the AptSIG Protocol

In Fig. 8 we show a concrete instantiation of the generic  $\Pi_{\text{aptSIG}}$  protocol from Fig. 7, called *aptSIG-BLS*. This instantiation uses tSIG implemented using threshold BLS as shown in Fig. 2 in Sect. 2.1, and the aPPSS shown in Fig. 4 in Sect. 3. Finally, the latter is instantiated with a specific OPRF protocol, 2HashDH [29], included in the full version of the paper. This concrete aptSIG protocol relies on authenticated channels between user  $U$  and each server  $S_i$  in initialization, an assumption we take throughout the paper. In addition, the initialization relies on a secure channel for  $U$ -to- $S_i$  communication, but secure channels can be implemented on top of authenticated channels using key exchange. Moreover, a typical application would use TLS to implement authenticated channels, which provides secure channels without any additional cost.

**Notation and Parameters.** Figure 8 assumes the following notation for public parameters: Security parameter  $l$ , threshold parameters  $t$ ,  $n$ ,  $t \leq n$ , field  $\mathbb{F} = GF(2^l)$ , cyclic group  $G$  of prime order  $p$ , bilinear map group  $\hat{G}$  of prime order  $\hat{p}$  and generator  $\hat{g}$ ; hash functions  $H_1, H_2, H_3, H_4$  with ranges  $G, \{0, 1\}^l, \{0, 1\}^{2l}, \hat{G}$ . Let  $\text{AE} = (\text{AuthEnc}, \text{AuthDec})$  be an Equivocable Authenticated Encryption.  $\text{auth}_{A \rightarrow B}\{m\}$  and  $\text{sec}_{A \rightarrow B}\{m\}$  stand for  $A$  sending message  $m$  to  $B$  via resp. authenticated and secure  $A \rightarrow B$  channel.

**Performance.** Our concrete aptSIG protocol is very practical: The initialization protocol takes 3 flows (after receiving OPRF replies the user can send all the remaining messages in one flow), and the signing protocol takes only 2 flows. Each server performs 2 exponentiations in both initialization and signing (one in a standard group, one in a group with a bilinear map), while the user performs  $O(n)$  exponentiations and one bilinear map. The protocol involves no server-to-server communication, and the bandwidth between user and each server is  $O(n)$ , but the only  $O(n)$ -sized message is a ciphertext vector  $\mathbf{e}$ , which can be stored more efficiently using error correction instead of replicating it on all servers, which reduces bandwidth to  $O(1)$  for  $t = O(n)$ . In the full version of the paper we show a simplified rendering of this protocol which highlights its simplicity and efficiency.

**Adding Robustness to aptSIG-BLS.** In the protocol in Fig. 8, user  $U$  chooses  $t + 1$  servers to interact with, and it aborts if any server misbehaves. Consequently, there is no guarantee that the protocol outputs a correct signature. To achieve guaranteed output, one needs to enhance the OPRF function with a *verifiable* OPRF [29], namely, where each server has a public OPRF verification key

Parameters: The notation and parameters are defined in text, on page 28.

Initialization for user U on input  $(sid, S_1, \dots, S_n, \mathbf{pw})$ :

1. Pick  $\alpha \leftarrow_{\mathbb{S}} \mathbb{Z}_p$ , set  $a = (H_1(\mathbf{pw}))^\alpha$ , and send  $((sid||i||0), a)$  to  $S_i$  for  $i \in [n]$ .
2. Receive  $\mathbf{auth}_{S_i \rightarrow U}\{a_i, b_i (= a^{k_i})\}$  for each  $S_i$ , abort if  $(a_i \neq a)$  for any  $i$ .
3. Pick  $s \leftarrow_{\mathbb{S}} \mathbb{F}$ , generate shares  $(s_1, \dots, s_n)$  as a  $(t, n)$ -secret-sharing of  $s$  over  $\mathbb{F}$ .  
Set  $\rho_i = H_2(\mathbf{pw}, b_i^{1/\alpha})$  and  $e_i = s_i \oplus \rho_i$  for all  $i \in [n]$ .
4. Set  $\mathbf{e} := (e_1, \dots, e_n)$ ,  $(C||\mathbf{sk}) := H_3(\mathbf{pw}, \mathbf{e}, s)$ , and  $\omega := (\mathbf{e}, C)$ .
5. Send  $\mathbf{auth}_{U \rightarrow S_i}\{(sid||i||1), \omega\}$  for all  $i \in [n]$ .
6. Pick  $v', v_0 \leftarrow_{\mathbb{S}} \mathbb{Z}_{\hat{p}}$ , set  $v = v_0 + v' \bmod \hat{p}$ , generate shares  $(v_1, \dots, v_n)$  as  $(t, n)$ -sharing of  $v'$  over  $\mathbb{Z}_{\hat{p}}$ . Send  $\mathbf{sec}_{U \rightarrow S_i}\{(sid||i), v_i\}$  for all  $i \in [n]$ .
7. Set  $\mathbf{V} = \hat{g}^v$  and  $\vec{\mathbf{V}} = (\mathbf{V}_1, \dots, \mathbf{V}_n)$  where  $\mathbf{V}_i = \hat{g}^{v_i}$  for every  $i \in [n]$ .  
Set  $\mathbf{aec}_U := \text{AE.AuthEnc}_{\mathbf{sk}}(\mathbf{U}, \mathbf{V}, \vec{\mathbf{V}}, v_0)$ , send  $\mathbf{auth}_{U \rightarrow S_i}\{(sid, \mathbf{aec}_U)\}$  for all  $i \in [n]$ . Output  $(\mathbf{ptsig.verifkey}, sid, \mathbf{V})$ .

Initialization for server S on input  $(sid, i, U)$ :

1. Set  $k \leftarrow_{\mathbb{S}} \mathbb{Z}_p$ , on  $((sid||i||0), a)$  from U, abort if  $a \notin G$ , else send  $\mathbf{auth}_{S \rightarrow U}\{a, a^k\}$ .
2. On message  $\mathbf{auth}_{U \rightarrow S}\{(sid||i||1), \omega\}$  from U, store  $(sid, i, \omega, k)$ .
3. Receive  $\mathbf{sec}_{U \rightarrow S}\{(sid||i), v_i\}$ , abort if  $v_i \notin \mathbb{Z}_{\hat{p}}$ . Save  $(sid, v_i)$ .
4. On  $\mathbf{auth}_{U \rightarrow S}\{sid, \mathbf{aec}_U\}$  save  $(sid, \mathbf{aec}_U)$ .

Signing for user U on input  $(sid, ssid, \mathbf{S} = \{S_1, \dots, S_{t+1}\}, \mathbf{pw}, m)$ :

1. Pick  $\alpha \leftarrow_{\mathbb{S}} \mathbb{Z}_p$ , set  $a = (H_1(\mathbf{pw}))^\alpha$ , send  $((sid, ssid, j), a)$  to  $S_j \in \mathbf{S}$ .
2. Given  $((sid, ssid, j), (b_j, i_j, \omega_j))$  and  $(sid, \mathbf{aec}_{U_j})$  from each  $S_j$ , set  $\phi_j = H_2(\mathbf{pw}, b_j^{1/\alpha})$  for  $j \in [t+1]$ . Abort if  $i_{j_1} = i_{j_2}$  or  $\omega_{j_1} \neq \omega_{j_2}$  for any  $j_1 \neq j_2$ .  
Otherwise set  $\rho_{i_j} := \phi_j$  for all  $j \in [t+1]$  and  $I := \{i_j | j \in [t+1]\}$ .
3. Parse any  $\omega_j$  as  $(\mathbf{e}, C)$  and  $\mathbf{e}$  as  $(e_1, \dots, e_n)$ . Set  $s_i := e_i \oplus \rho_i$  for each  $i \in I$ .
4. Recover  $s$  and the shares  $s_i$  for  $i \notin I$  by interpolating points  $(i, s_i)$  for  $i \in I$ .
5. Parse  $H_3(\mathbf{pw}, \mathbf{e}, s)$  as  $(C' || \mathbf{sk})$ . Abort if  $C' \neq C$ .
6. Abort if  $\mathbf{aec}_{U_{j_1}} \neq \mathbf{aec}_{U_{j_2}}$  for any  $j_1, j_2 \in [t+1]$ , else set  $\mathbf{aec}_U$  to any  $\mathbf{aec}_{U_j}$ . Abort if  $\text{AE.AuthDec}_{\mathbf{sk}}(\mathbf{aec}_U) = \perp$ , else parse  $\text{AE.AuthDec}_{\mathbf{sk}}(\mathbf{aec}_U)$  as  $(\mathbf{U}, \mathbf{V}, \vec{\mathbf{V}}, v_0)$ .
7. On messages  $(j, \sigma_j)$  from each  $S_j \in \mathbf{S}$  if  $e(g, \sigma_j) \neq e(\mathbf{V}_j, H_4(m))$  for any  $j \in [t+1]$ , output  $(\mathbf{ptsig.finsign}, sid, ssid, m, \perp)$ . Else compute  $\sigma := \sigma_0 \cdot (\prod_{j \in \mathbf{S}} (\sigma_j)^{\lambda_i})$ , where  $\sigma_0 = H_4(m)^{v_0}$  and  $\lambda_i$ 's are Lagrange interpolation coefficients corresponding to the set of indexes in  $\mathbf{S}$  corresponding to  $\mathbf{S}$ .
8. Output  $(\mathbf{ptsig.finsign}, sid, ssid, m, \sigma)$ .

Signing for server S on input  $(sid, ssid, U, m)$ :

1. Given  $((sid, ssid, j), a)$  from U, abort if  $a \notin G$  or S does not hold records  $(sid, i, \omega, k)$ ,  $(sid, v_i)$  and  $(sid, \mathbf{aec}_U)$  with the matching  $sid$ .
2. Otherwise set  $b := a^k$  and send  $((sid, ssid, j), (b, i, \omega))$ ,  $(sid, \mathbf{aec}_U)$  to U.
3. Send  $(i, \sigma)$  to U, where  $\sigma := H_4(m)^{v_i}$ .

**Fig. 8.** *aptSIG-BLS*: an aptSIG protocol instantiated with T-BLS and aPSS of Fig. 4 with  $\mathcal{H}HashDH$  OPRF. The aPSS sub-protocol is marked in gray .

that is provided to the user at initialization and included in the vector  $\omega$ . In particular, the OPRF construction can be made verifiable (see [29]) by setting each server's public key to  $g^k$  where  $k$  is the server's OPRF key and where verification is implemented via a non-interactive ZK proof of equality of dlog. In this case,  $\mathcal{U}$  can run the aPSS protocol with any subset of  $t+1$  or more servers that sent the same  $\omega$  value. If reconstruction succeeds, the user has correct keying material, including the public keys to verify individual BLS signatures by the servers (and discard invalid signatures). If reconstruction fails, a new (disjoint) set of  $t+1$  or more servers with same value  $\omega$  is chosen by  $\mathcal{U}$  and the process is repeated. It is guaranteed that if  $\mathcal{U}$  has undisturbed connectivity to  $t+1$  honest servers, the correct signature  $\sigma$  on message  $m$  will be produced. The above process repeats for at most  $\lceil n/(t+1) \rceil$  times, hence it is efficient even with dishonest majority.

**Adding PFS Security to aptSIG-BLS.** In the protocol in Fig. 8, server  $S_i$  in step 3 of the signing phase sends its partial signature  $\sigma_i$  without a proof that  $\mathcal{U}$  knows the correct password  $\text{pw}$  and wants to sign  $m$ . This enables the adversary to gather partial signatures on a message  $m$  without prior knowledge of  $\text{pw}$ , and then complete these to the full signature if it compromises password  $\text{pw}$  in the future. However, we can prevent this attack and guarantee Perfect Forward Secrecy (PFS). This extension is sketched at the end of Sect. 4.1, and is fully described in the full version of the paper. The PFS-version of the fully instantiated protocol aptSIG-BLS is deferred to the full version of the paper.

**Acknowledgments.** *Stefan Dziembowski, Pawel Kedzior, Chan Nam Ngo:* This work is part of a project that received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grants PROCONTRA-885666). *Stefan Dziembowski* was also partly supported by the Polish NCN Grant 2019/35/B/ST6/04138 and the Nicolaus Copernicus Polish-German Research Award 2020 COP/01/2020. *Stanislaw Jarecki:* This work was supported by NSF SaTC TTP award 2030575. *Hugo Krawczyk:* This work was done while the author was at the Algorand Foundation. *Chan Nam Ngo:* The majority of this work was done while the author was with the University of Warsaw, Poland.

## References

1. Agrawal, S., Miao, P., Mohassel, P., Mukherjee, P.: PASTA: PASsword-based threshold authentication. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 2042–2059. ACM Press (Oct 2018)
2. Arapinis, M., Gkaniatsou, A., Karakostas, D., Kiayias, A.: A formal treatment of hardware wallets. In: Goldberg, I., Moore, T. (eds.) Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers. Lecture Notes in Computer Science, vol. 11598, pp. 426–445. Springer (2019). [https://doi.org/10.1007/978-3-030-32101-7\\_26](https://doi.org/10.1007/978-3-030-32101-7_26), [https://doi.org/10.1007/978-3-030-32101-7\\_26](https://doi.org/10.1007/978-3-030-32101-7_26)
3. Aumasson, J., Hamelink, A., Shlomovits, O.: A survey of ECDSA threshold signing. IACR Cryptol. ePrint Arch. p. 1390 (2020), <https://eprint.iacr.org/2020/1390>

4. Bacho, R., Loss, J.: On the adaptive security of the threshold BLS signature scheme. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 193–207. ACM Press (Nov 2022). <https://doi.org/10.1145/3548606.3560656>
5. Bagherzandi, A., Jarecki, S., Saxena, N., Lu, Y.: Password-protected secret sharing. In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) ACM CCS 2011. pp. 433–444. ACM Press (Oct 2011)
6. Baum, C., Frederiksen, T., Hesse, J., Lehmann, A., Yanai, A.: Pesto: Proactively secure distributed single sign-on, or how to trust a hacked server. In: 2020 IEEE European Symposium on Security and Privacy (EuroSP). pp. 587–606 (2020)
7. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In: Desmedt, Y. (ed.) Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6–8, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2567, pp. 31–46. Springer (2003). [https://doi.org/10.1007/3-540-36288-6\\_3](https://doi.org/10.1007/3-540-36288-6_3), [https://doi.org/10.1007/3-540-36288-6\\_3](https://doi.org/10.1007/3-540-36288-6_3)
8. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. *J. Cryptol.* **17**(4), 297–319 (2004). <https://doi.org/10.1007/s00145-004-0314-9>, <https://doi.org/10.1007/s00145-004-0314-9>
9. Boyd, C.: Digital multisignatures. *Cryptography and Coding* (1986)
10. Camenisch, J., Lehmann, A., Neven, G., Samelin, K.: Virtual smart cards: How to sign with a password and a server. In: Zikas, V., De Prisco, R. (eds.) SCN 16. LNCS, vol. 9841, pp. 353–371 (Aug / Sep 2016)
11. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14–17 October 2001, Las Vegas, Nevada, USA. pp. 136–145. IEEE Computer Society (2001). <https://doi.org/10.1109/SFCS.2001.959888>, <https://doi.org/10.1109/SFCS.2001.959888>
12. Canetti, R.: Universally composable signatures, certification and authentication. *Cryptology ePrint Archive*, Report 2003/239 (2003), <https://eprint.iacr.org/2003/239>
13. Canetti, R., Gennaro, R., Goldfeder, S., Makriyannis, N., Peled, U.: UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1769–1787. ACM Press (Nov 2020)
14. Castagnos, G., Catalano, D., Laguillaumie, F., Savasta, F., Tucker, I.: Bandwidth-efficient threshold EC-DNA. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111 (May 2020)
15. Das, P., Erwig, A., Faust, S., Loss, J., Riahi, S.: Bip32-compatible threshold wallets. *IACR Cryptol. ePrint Arch.* p. 312 (2023), <https://eprint.iacr.org/2023/312>
16. Das, S., Ren, L.: Adaptively secure BLS threshold signatures from DDH and co-CDH. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part VII. LNCS, vol. 14926, pp. 251–284. Springer, Cham (Aug 2024)
17. Desmedt, Y.: Society and group oriented cryptography: A new concept. In: Pomerance, C. (ed.) CRYPTO’87. LNCS, vol. 293 (Aug 1988)
18. Desmedt, Y., Frankel, Y.: Threshold cryptosystems. In: Brassard, G. (ed.) CRYPTO’89. LNCS, vol. 435 (Aug 1990)
19. Desmedt, Y., Frankel, Y.: Shared generation of authenticators and signatures (extended abstract). In: Feigenbaum, J. (ed.) CRYPTO’91. LNCS, vol. 576 (Aug 1992)

20. Doerner, J., Kondi, Y., Lee, E., shelat, a.: Threshold ECDSA from ECDSA assumptions: The multiparty case. In: 2019 IEEE Symposium on Security and Privacy. IEEE Computer Society Press (May 2019)
21. Dziembowski, S., Jarecki, S., Kedzior, P., Krawczyk, H., Ngo, C.N., Xu, J.: Password-protected threshold signatures. Cryptology ePrint Archive, Paper number TBD (2024), TBD
22. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62 (Aug 2018). [https://doi.org/10.1007/978-3-319-96881-0\\_2](https://doi.org/10.1007/978-3-319-96881-0_2)
23. Ganesan, R.: Yaksha: augmenting kerberos with public key cryptography. In: Proceedings of the Symposium on Network and Distributed System Security. pp. 132–143 (1995)
24. Gennaro, R., Goldfeder, S.: Fast multiparty threshold ECDSA with fast trustless setup. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. ACM Press (Oct 2018)
25. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptol.* **20**(1), 51–83 (2007). <https://doi.org/10.1007/s00145-006-0347-3>, <https://doi.org/10.1007/s00145-006-0347-3>
26. Gentry, C., MacKenzie, P.D., Ramzan, Z.: A method for making password-based key exchange resilient to server compromise. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings. Lecture Notes in Computer Science, vol. 4117, pp. 142–159. Springer (2006). [https://doi.org/10.1007/11818175\\_9](https://doi.org/10.1007/11818175_9), [https://doi.org/10.1007/11818175\\_9](https://doi.org/10.1007/11818175_9)
27. Gjøsteen, K., Thuen, Ø.: Password-based signatures. In: Petkova-Nikova, S., Pashalis, A., Pernul, G. (eds.) Public Key Infrastructures, Services and Applications. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
28. Gu, Y., Jarecki, S., Kedzior, P., Nazarian, P., Xu, J.: Threshold PAKE with security against compromise of all servers. In: Advances in Cryptology – ASIACRYPT 2024 (2024)
29. Jarecki, S., Kiayias, A., Krawczyk, H.: Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II. Lecture Notes in Computer Science, vol. 8874, pp. 233–253. Springer (2014). [https://doi.org/10.1007/978-3-662-45608-8\\_13](https://doi.org/10.1007/978-3-662-45608-8_13), [https://doi.org/10.1007/978-3-662-45608-8\\_13](https://doi.org/10.1007/978-3-662-45608-8_13)
30. Jarecki, S., Kiayias, A., Krawczyk, H., Xu, J.: Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online). In: 2016 IEEE European Symposium on Security and Privacy (EuroSP). pp. 276–291 (2016). <https://doi.org/10.1109/EuroSP.2016.30>
31. Jarecki, S., Kiayias, A., Krawczyk, H., Xu, J.: TOPPSS: cost-minimal password-protected secret sharing based on threshold OPRF. In: Gollmann, D., Miyaji, A., Kikuchi, H. (eds.) Applied Cryptography and Network Security - 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10355, pp. 39–58. Springer (2017). [https://doi.org/10.1007/978-3-319-61204-1\\_3](https://doi.org/10.1007/978-3-319-61204-1_3), [https://doi.org/10.1007/978-3-319-61204-1\\_3](https://doi.org/10.1007/978-3-319-61204-1_3)
32. Jarecki, S., Krawczyk, H., Xu, J.: OPAQUE: an asymmetric PAKE protocol secure against pre-computation attacks. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the

- Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III. Lecture Notes in Computer Science, vol. 10822, pp. 456–486. Springer (2018). [https://doi.org/10.1007/978-3-319-78372-7\\_15](https://doi.org/10.1007/978-3-319-78372-7_15), [https://doi.org/10.1007/978-3-319-78372-7\\_15](https://doi.org/10.1007/978-3-319-78372-7_15)
33. Lindell, Y., Nof, A.: Fast secure multiparty ECDSA with practical distributed key generation and applications to cryptocurrency custody. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. ACM Press (Oct 2018)
  34. MacKenzie, P.D., Reiter, M.K.: Networked cryptographic devices resilient to capture. In: 2001 IEEE Symposium on Security and Privacy. pp. 12–25. IEEE Computer Society Press (May 2001)
  35. MacKenzie, P.D., Shrimpton, T., Jakobsson, M.: Threshold password-authenticated key exchange. *J. Cryptol.* **19**(1), 27–66 (2006). <https://doi.org/10.1007/s00145-005-0232-5>, <https://doi.org/10.1007/s00145-005-0232-5>
  36. McQuoid, I., Rosulek, M., Xu, J.: How to obfuscate MPC inputs. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part II. LNCS, vol. 13748, pp. 151–180. Springer, Cham (Nov 2022)
  37. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140 (Aug 1992)
  38. Wikström, D.: Universally composable DKG with linear number of exponentiations. In: Blundo, C., Cimato, S. (eds.) Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8–10, 2004, Revised Selected Papers. Lecture Notes in Computer Science, vol. 3352, pp. 263–277. Springer (2004). [https://doi.org/10.1007/978-3-540-30598-9\\_19](https://doi.org/10.1007/978-3-540-30598-9_19), [https://doi.org/10.1007/978-3-540-30598-9\\_19](https://doi.org/10.1007/978-3-540-30598-9_19)
  39. Xu, S., Sandhu, R.S.: Two efficient and provably secure schemes for server-assisted threshold signatures. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 355–372 (Apr 2003)



# Strongly Secure Universal Thresholdizer

Ehsan Ebrahimi<sup>1</sup>(✉) and Anshu Yadav<sup>2</sup>

<sup>1</sup> University of Luxembourg, Esch-sur-Alzette, Luxembourg  
ehsan.ebrahimi@uni.lu

<sup>2</sup> Institute of Science and Technology, Klosterneuburg, Austria

**Abstract.** A *universal thresholdizer* (UT), constructed from a threshold fully homomorphic encryption by Boneh *et. al* , Crypto 2018, is a general framework for universally thresholdizing many cryptographic schemes. However, their framework is insufficient to construct strongly secure threshold schemes, such as threshold signatures and threshold public-key encryption, etc.

In this paper, we strengthen the security definition for a *universal thresholdizer* and propose a scheme which satisfies our stronger security notion. Our UT scheme is an improvement of Boneh *et. al* 's construction at the level of threshold fully homomorphic encryption using a key homomorphic pseudorandom function. We apply our strongly secure UT scheme to construct strongly secure threshold signatures and threshold public-key encryption.

**Keywords:** Universal Thresholdizer · Threshold Signature · Threshold FHE · Threshold PKE

## 1 Introduction

Threshold cryptography [DF89] refers to distributing a privileged operation that requires a secret key, like signing in a signature scheme and decryption in an encryption key, among  $n$  parties so that any  $t$  of them can collaborate to perform the final computation. Distributing shares of the secret key between multiple parties makes the system fault-tolerant. The reasons are: 1) A corrupted number of parties below the threshold value are not able to evaluate the cryptographic primitive. 2) The system would perform correctly, even in the presence of few corrupt parties, if honest parties beyond the threshold value participate in the evaluation. Due to the importance of protecting the secret key, almost all the cryptographic primitives have their corresponding threshold system. Examples are threshold public-key encryption schemes [CG99], threshold signature schemes [Sho00], threshold pseudorandom functions [NPR99], threshold symmetric encryption [AMMR18], threshold message authentication codes [MPS+02], etc.

In a non-interactive threshold scheme, each party computes a partial output completely independently of others and then any  $t$  partial outputs can be publicly combined to get the final output. A natural notion of security then says that no



polynomial time adversary must be able to compute the final output even if it is given up to  $t - 1$  partial outputs. Most of the papers on threshold cryptography model this by allowing the adversary to get the partial secret keys of  $t - 1$  parties. Ideally, the adversary should be allowed to choose the  $t - 1$  parties adaptively. However, achieving adaptive security is hard, and hence, generally the security definitions restrict the adversary to output the  $t - 1$  parties in the beginning. In addition, they also disallow the adversary to ask the partial outputs on challenge inputs (for e.g., partial signature on challenge message in threshold signature and partial decryption of challenge ciphertext in CCA security of threshold PKE scheme) [GWW+13, dPKM+24, Bol02, BGG+18]. This notion of security was strengthened by Bellare, *et. al* [BCK+22] in the context of threshold signatures, where the adversary is now allowed to output a forgery on one of the messages for which it has already asked a partial signature. In more detail, let  $c$  be the number of parties whose keys are given to the adversary. Then the adversary is allowed to output a forgery on any message for which it has received at most  $t - 1 - c$  partial signatures<sup>1</sup>. They further showed that some of the well-known signature schemes, BLS [BLS04, Bol03] and FROST [KG20] can, in fact, be proven secure in their stronger security definition. BLS and FROST are based on classical assumptions based on discrete logarithm problem and is thus not secure against a quantum adversary.

Motivated by Bellare *et. al* 's work, we started with the question - is the only lattice based non-interactive threshold signature scheme of Boneh *et. al* [BGG+18] (referred to as BGGJKRS from now on) secure in the stronger definition given by Bellare *et. al* ? We found that it is hard to prove the stronger security for BGGJKRS in its current form. We discuss the issues in the technical overview. We observed that the stronger definition can in fact be viewed as an adaptive version of the current definition, which explains the difficulty in proving it. We provide a detailed discussion on adaptive nature of the Bellare *et. al* 's stronger security notion in technical overview.

We then improved BGGJKRS construction using key homomorphic pseudo-random functions (KHPRF), which are standard PRF, with additional property that for all  $K_1, K_2$ , for all inputs  $x$ ,  $\text{PRF}(K_1, x) + \text{PRF}(K_2, x) = \text{PRF}(K_1 + K_2, x)$ . We then show that our improved construction satisfies the stronger security notion of Bellare *et. al* .

Boneh *et. al* constructed the threshold signature from a tool called universal thresholdizer (UT), which they defined and built from lattice based assumptions. The universal thresholdizer can be used as a compiler to thresholdize any cryptographic primitive and is itself built from threshold FHE, which again, the authors define and construct from any "special" FHE in the same paper. We realized that our improvement in threshold signature scheme can actually be implemented at the level of TFHE so that it improves its security, which in turn, improves the security of universal thresholdizer built from TFHE in a way that helps in achieving the stronger security for any threshold cryptographic

---

<sup>1</sup> Bellare *et. al* defined different levels of security improvements. However, in this work, we focus only on their TS-UF-1 definition.

primitive that uses UT as the thresholdizer. In the main body of this paper, we define the desired stronger notion of security for TFHE and UT and provide a construction for TFHE that achieves this notion. We then show that the universal thresholdizer built from our TFHE achieves the desired security. We finally show applications of stronger universal thresholdizer to construct threshold signatures and CCA secure PKE with the stronger security. However, in the technical overview, we describe the challenges and our ideas using the specific case of threshold signatures, because some of the ideas, especially the adaptivity perspective of the stronger security definition is more clear when viewed at the applications level, instead of TFHE or UT.

### Our Contributions:

- We strengthen the security definition of universal thresholdizer (UT) in [BGG+18], which is needed to prove the stronger security of threshold cryptoprimitives built using the UT. In turn, we define stronger security property for threshold FHE (TFHE) needed to prove the stronger security of UT.
- We improve the construction of TFHE in [BGG+18] to achieve the stronger security property, we define.
- Using our TFHE, we get the first lattice based construction of non-interactive threshold signature with the stronger security as defined in [BCK+22, BTZ22].
- Along the lines of [BCK+22], we define a stronger security notion for CCA-PKE, and show that the constructions in [BGG+18] satisfy this security if the universal thresholdizer UT satisfies stronger security.
- In [ASY22], Agrawal *et. al* constructed partially adaptive threshold signatures in random oracle model, that allows the adversary to issue key queries (all at once) in the middle of the game. We combine our technique with Agrawal *et. al* 's to construct TFHE (and its applications) with stronger security while also allowing the adversary to issue key queries (all at once) in the middle of the game.

## 1.1 Stronger Security from Adaptivity Perspective

A natural notion of security for  $t$ -out-of- $n$  threshold signatures says that any PPT adversary  $\mathcal{A}$  should not be able to generate a signature on any message  $m^*$  even if it gets (i) partial signatures on  $m^*$  from up to any  $t - 1$  parties and (ii) complete (or any number of partial) signatures on messages  $m \neq m^*$ . Item (ii) is easy to provide and considered by all versions of unforgeability definition in the literature. For item (i),  $\mathcal{A}$  can obtain partial signatures on  $m^*$  in two ways - (i) get the partial signing keys from the  $t - 1$  parties (ii) only get the partial signatures (but not the keys) on  $m^*$  from these parties. Clearly, the adversary gets more power in the first case. In an ideal situation, the adversary can adaptively decide which  $t - 1$  parties to corrupt. However, this natural notion of security (mostly referred to as adaptive unforgeability) is hard to achieve and hence, most of the constructions in this area consider a selective notion of unforgeability where the adversary must decide the set of corrupted parties in the beginning. There is

also a notion of *partial adaptivity*, where the adversary can choose the set of corrupted parties in the middle of the unforgeability game (i.e. after seeing some (partial) signatures on other messages), but the entire set of corrupted parties must be declared together [ASY22]. Since once the adversary has decided the set of corrupted parties, it is always beneficial for it to ask the signing keys rather than just the partial signatures, these definitions assume that the adversary gets the partial signing keys from  $t - 1$  parties, and is not allowed to issue any partial signing query on the challenge message  $m^*$ . However, we can also consider partial adaptivity in an orthogonal direction, where the adversary divides the set of corrupted parties as *fully corrupted*, for which it asks the signing keys and *semi-corrupted* from which it only asks the partial signature on  $m^*$ . The adversary must output the set of fully corrupted parties selectively, i.e., in the beginning of the game, but can decide the semi-corrupted parties adaptively throughout the game. This is the notion considered in [BCK+22, BTZ22] under stronger security definition which they call as TS-UF-1. Indeed we can define security properties that combine adaptivity in both the directions. In Fig. 1, we present different security properties that can be thus obtained, and relation between them.

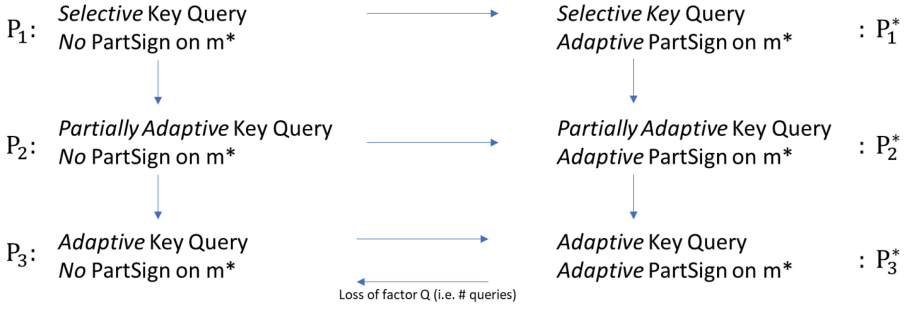
Between  $P_2$  and  $P_1^*$ , we could not show either of them implied by the other. However, this is not surprising since both  $P_1^*$  and  $P_2$  improve  $P_1$  in different directions.

What is interesting is that even  $P_3$  does not seem to imply  $P_1^*$  except with a loss of factor  $Q$ , where  $Q$  is the number of signing queries. This can be understood as follows. Let TS be any threshold signature scheme and  $\mathcal{A}_{P_1^*}$  be an adversary against  $P_1^*$  security of TS. We want to show that if  $\mathcal{A}_{P_1^*}$  exists then there exists an adversary  $\mathcal{A}_{P_3}$  against  $P_3$  security of TS. The difficulty in the reduction comes when  $\mathcal{A}_{P_1^*}$  issues a signing query  $(m, i)$  such that  $m = m^*$ . In this situation,  $\mathcal{A}_{P_3}$  must not ask the partial signature from  $P_3$  challenger. Instead it should ask for  $\text{fsk}_i$  (the partial signing key of party  $i$ ) and compute  $\text{PartSign}(i, \text{fsk}_i, m)$  itself. But the problem is that  $\mathcal{A}_{P_3}$  does not know when  $m = m^*$ . So,  $\mathcal{A}_{P_3}$  guesses  $m^*$  as follows: let  $Q_s$  be the number of different messages queried for partial signatures. Then  $Q_s \leq Q$ , where  $Q$  is the total number of signing queries. Guess an index  $k \in [Q_s + 1]$  and assume the  $k$ -th unique message as  $m^*$ . Then, for each signing query  $(m, i)$  with  $m \neq m^*$  from  $\mathcal{A}_{P_1^*}$ , the reduction forwards the query to  $P_3$  challenger. For  $m = m^*$ , the reduction queries for  $\text{fsk}_i$  and computes the partial signature itself. Thus,  $P_3$  implies  $P_1^*$ , but with a loss of factor  $Q_s \leq Q$ .

We summarize all the discussed notions in Fig. 1. An arrow in the picture indicates a path to strengthen a notion, i.e., it is not an implication.

## 1.2 Technical Overview

Our construction uses the BGGJKRS construction, which does not satisfy the stronger security notion, as the base and builds upon it to get the stronger security. We first recall the BGGJKRS construction for threshold signature.



**Fig. 1.** Threshold Signature security with different level of adaptivity. For any two properties,  $P_i, P_j, P_i \rightarrow P_j$  means that  $P_j$  is stronger than (at least as strong as)  $P_i$ .

**Recap of BGGJKRS Threshold Signature.** The BGGJKRS threshold signature scheme is a round optimal (non-interactive) scheme built from a “universal thresholdizer” that can thresholdize a number of cryptographic primitives. The thresholdizer is built from a thresholdized version of any “special” fully homomorphic encryption (FHE) where the decryption of any ciphertext  $ct$ , using a key  $fsk$ , is a linear operation as  $\langle ct, fsk \rangle$ , which gives  $\lfloor q/2 \rfloor m + e$ , followed with rounding which gives  $m$ . Here,  $q$  is the working modulus and  $e$  is a “small” error and depends on the LWE assumption on which FHE is built. In the thresholdized version the decryption key  $fsk$  is  $t$ -out-of- $n$  secret shared between  $n$  parties. To decrypt a ciphertext  $ct$ , each party can compute a partial decryption of  $ct$  using its own key share  $fsk_i$  and then any  $t$  partial decryptions can be (publicly) combined to get a complete decryption. The BGGJKRS construction for threshold signature  $TS = (TS.Setup, TS.PartSign, TS.Combine)$  uses (i) a linear secret sharing scheme  $SS$ , where the individual shares are combined through a linear process ( $SS.Combine$ ) to get the full secret, (ii) a fully homomorphic encryption scheme FHE, where the decryption algorithm is a linear operation<sup>2</sup>, and (iii) a signature scheme  $Sig$ . Then the construction is:

- $TS.Setup()$ : generates FHE keys  $(fpk, fsk)$ , signing keys  $(svk, ssk)$ , and encrypts  $ssk$  as  $ct = FHE.Enc(ssk)$ . It then uses  $SS$  to secret share  $fsk$  among the  $n$  parties as  $\{fsk_i\}_{i \in [n]}$  for  $t$ -out-of- $n$  threshold access structure. Finally, it sets  $pp = (fpk, svk, ct), \{sk_i = fsk_i\}_{i \in [n]}, vk = svk$ .
- $TS.PartSign(i, sk_i, m)$ : firstly computes encryption of signature,  $\sigma_m$ , on  $m$  using FHE. Eval as  $ct_{\sigma_m} = FHE.Eval(Sig.Sign_m, ct)$  and returns partial decryption of  $ct_{\sigma_m}$  as  $\sigma_{m,i} = \langle ct_{\sigma_m}, fsk_i \rangle + noise$ . Here,  $Sig.Sign_m$  is the signing circuit of  $Sig$  with message  $m$  being hardwired.

<sup>2</sup> For ciphertext  $ct \in \mathbb{Z}_q^\lambda$  and decryption key  $fsk \in \mathbb{Z}_q^\lambda$ ,  $FHE.Decrypt(fsk, ct)$  first computes  $\langle ct, fsk \rangle$ , which gives  $\lfloor q/2 \rfloor m + e$ , followed by rounding which outputs  $m$ .

- $\text{TS.Combine}(S, \{\sigma_{m,i}\}_{i \in S})$ : For  $|S| \geq t$ , the TS combine algorithm first runs the  $\text{SS.Combine}$  algorithm on the partial signatures and then performs rounding to get the signature as  $\sigma_m = \text{round}(\text{SS.Combine}(\{\sigma_{m,i}\}_{i \in S}))$ .

**Security Sketch for BGGJKRS Construction.** We first briefly recall the security game. In the beginning of the game, the adversary is given the partial signing keys from up to  $t - 1$  parties of its choice. In addition, the adversary can adaptively issue partial signing queries of the form  $(m, i)$  to receive partial signature  $\sigma_{m,i}$  on  $m$  from party  $i$ . In the end, to win the game, the adversary must output a forgery  $(m^*, \sigma^*)$  on a message  $m^*$  for which no partial signature was requested.

For the ease of presentation, let us consider a simple case of 2-out-of-2 access structure. Thus, the FHE decryption key  $\text{fsk}$  is secret shared between the two parties as: pick a random  $\text{fsk}_1$  and set  $\text{fsk}_2 = \text{fsk} - \text{fsk}_1$ . Wlog, we assume that the adversary asks the signing key for the first party<sup>3</sup>.

The security is through a sequence of hybrids, where as usual, the initial hybrid ( $H_0$ ) is the real game. Then in the next hybrid ( $H_1$ ), for any queried message  $m$ , the challenger computes the partial signatures corresponding to the honest party  $P_2$  without using its signing key  $\text{fsk}_2$ . Instead,  $\sigma_{m,2}$  is computed as  $\lfloor q/2 \rfloor \sigma_m - \langle \text{ct}_{\sigma_m}, \text{fsk}_1 \rangle + \text{noise}$ , while  $\sigma_{m,1}$  is computed honestly as  $\langle \text{ct}_{\sigma_m}, \text{fsk}_1 \rangle + \text{noise}$ <sup>4</sup>. This does not change the adversary’s view because of linearity of FHE.  $\text{Decrypt}$ . The purpose behind this step is to use the shares of FHE key  $\text{fsk}$  only from an “invalid” set of parties (i.e. a set having at most  $t - 1$  parties) so that, then, in the next hybrid ( $H_2$ ), the setup algorithm can use zero vector instead of  $\text{fsk}$  to generate the partial signing keys  $\text{fsk}_1, \text{fsk}_2$ , using the security of the secret sharing scheme. At this stage,  $\text{fsk}$  is not used at all, and hence in the next hybrid ( $H_3$ ), using FHE security,  $\text{ct}$  in the setup is changed from  $\text{ct} = \text{FHE.Encrypt}(\text{ssk})$  to  $\text{ct} = \text{FHE.Encrypt}(0)$ . Finally, at this stage  $\text{ssk}$  is not used and we can use the security of underlying signature scheme to argue that the adversary cannot output a forgery in this hybrid, which implies that the adversary cannot generate a forgery in the real world as well.

The reduction to Sig security goes as follows. After getting  $\text{svk}$  from the Sig challenger the reduction algorithm samples FHE keys  $(\text{fpk}, \text{fsk})$  and discards  $\text{fsk}$ . It secret shares 0 into  $\text{fsk}_1$  and  $\text{fsk}_2$ , encrypts 0 as  $\text{ct} = \text{FHE.Encrypt}(0)$  and sends  $\text{pp} = (\text{fpk}, \text{ct})$  and  $\text{fsk}_1$  as signing key for party 1. Whenever the adversary issue signing query on any message  $m$  from party 2, the reduction algorithm computes  $\text{ct}_{\sigma_m}$  and sends a signing query on  $m$  to Sig challenger and gets  $\sigma_m$ . It then computes  $\sigma_{m,2} = \sigma_m - \langle \text{ct}_{\sigma_m}, \text{fsk}_1 \rangle + \text{noise}$ .

In the end when the adversary outputs a forgery  $(m^*, \sigma_{m^*})$ , the reduction forwards it to the Sig challenger. Since the adversary is not allowed to query

<sup>3</sup> Since it is in the best interest of the adversary to get keys from the maximum possible number of parties, the security definition assumes that the adversary gets keys from  $t - 1$  parties.

<sup>4</sup>  $\text{noise}$  is added to hide the FHE error  $e$ , but is not the main focus of current discussion.

partial signatures on  $m^*$ , the reduction never asks signature on  $m^*$  from the Sig challenger and hence,  $(m^*, \sigma_{m^*})$  is a valid forgery against the Sig challenger.

**Problem in Proving Stronger Security.** In the stronger security, the adversary is allowed to query partial signature on  $m^*$  from up to  $g$  parties, where  $g = t - 1 - c$  and  $c$  is the number of parties for which the signing key is obtained. To better understand the difficulty in proving the stronger security in BGGJKRS, let us consider a scenario where the adversary does not issue any key query. Following the proof strategy as before, while answering partial signing queries, the challenger would want to make sure that it uses only either  $\text{fsk}_1$  or  $\text{fsk}_2$  but not both. The challenger needs to decide, which one? But, since now partial signature queries on  $m^*$  are also allowed, when a signing query  $(m, i)$  for  $i \in \{1, 2\}$  is received, the challenger needs to be careful in using  $\sigma_m$  because in the end, if  $m^* = m$ , the reduction against the Sig challenger in the last hybrid fails. This is so because whenever the challenger needs  $\sigma_m$  to reply a partial signing query on  $m$  in ( $H_3$  in the proof of BGGJKRS scheme above), the reduction against Sig gets it from the Sig challenger, and in this case, the reduction cannot output a forgery on  $m$ . Let us understand the challenger's dilemma with the following example:

Suppose the adversary issues the first signing query as  $(m_1, 1)$ . Now the challenger needs to decide whether to use  $\text{fsk}_1$  to compute  $\sigma_{m_1,1}$  or not. Let us analyze both the options and show that both the choices can go wrong:

Option 1:

- The challenger uses  $\text{fsk}_1$  to compute  $\sigma_{m_1,1}$ .
- The adversary issues next query as  $(m_2, 2)$ . Now since  $\text{fsk}_1$  is already used, the challenger cannot use  $\text{fsk}_2$  to generate  $\sigma_{m_2,2}$ . Hence, it computes  $\sigma_{m_2,2}$  as  $\lfloor q/2 \rfloor \sigma_{m_2} - \langle \text{ct}_{\sigma_{m_2}}, \text{fsk}_1 \rangle + \text{noise}$ .
- The adversary outputs a forgery  $(m_2, \sigma_{m_2})$ . Observe that this is a valid forgery from the TS adversary since it has queried partial signature on  $m_2$  from party 2 only. But since the challenger used  $\sigma_{m_2}$  to compute the partial signature,  $(m_2, \sigma_{m_2})$  cannot be returned as a forgery to the Sig challenger.

Option 2:

- The challenger uses  $\text{fsk}_2$  to compute  $\sigma_{m_1,1}$  as  $\sigma_{m_1,1}$  as  $\lfloor q/2 \rfloor \sigma_{m_1} - \langle \text{ct}_{\sigma_{m_1}}, \text{fsk}_2 \rangle + \text{noise}$ .
- The adversary outputs a forgery  $(m_1, \sigma_{m_1})$ . Observe that this is a valid forgery from the TS adversary since it has queried partial signature on  $m_1$  from party 1 only. But since the challenger used  $\sigma_{m_1}$  to compute the partial signature,  $(m_1, \sigma_{m_1})$  cannot be returned as a forgery to the Sig challenger.

*Remark 1.* In the toy example above, one may be tempted to use guessing, and that would indeed work here. But guessing becomes hard for general case of  $t$ -out-of- $n$  access structure with neither  $n - t$  nor  $t - c$  being a constant, where  $c$  is the number of parties for which the adversary asks the signing keys.

**Our Solution:** From the above arguments we can derive the following wishful strategy for the challenger (i) use the secret shares of  $\text{fsk}$  from at most  $t-1$  parties (as before), (ii) for any signing query  $(m, i)$  do not use  $\sigma_m$  till  $|S_m \cup S^*| \leq t-1$ , where  $S^*$  is the set of parties for which the adversary asks the signing keys and  $S_m = \{j : (m, j) \text{ has been queried so far (including the query } (m, i))\}$ .

*Attempt 1:* For each query  $(m, i)$  simply return a random value + noise till  $|S^* \cup S_m| \leq t-1$  and argue that since  $\text{fsk}_i$  is random,  $\langle \text{ct}_{\sigma_m}, \text{fsk}_i \rangle$  is also random. But we can observe that this strategy immediately fails if the adversary issues partial signing query from party  $i$  for  $Q$  different messages for  $Q > |\text{fsk}_i|$ .

*Solution:* We use the same strategy as before, where for each query  $(m, i)$ , the challenger simply returns a random value + noise till  $|S^* \cup S_m| \leq t-1$ . But now, we use a different argument to argue indistinguishability from the honestly computed value using  $\text{fsk}_i$ . In more detail, let us consider the previous toy example of 2-out-of-2 access structure. Let the adversary issues first signing query as  $(m_1, 1)$ . The challenger returns

$$\sigma_{m_1,1} = r_{m_1,1} + \text{noise},$$

where  $r_{m_1,1}$  is uniformly random. Then if and whenever the adversary issues a signing query  $(m_1, 2)$ , the challenger returns

$$\sigma_{m_1,2} = \lfloor q/2 \rfloor \sigma_{m_1} - r_{m_1,1} + \text{noise}.$$

Notice that this time it is safe to use  $\sigma_{m_1}$  because party 1 and party 2 together form a valid party set and hence the challenger knows that  $m_1$  cannot be  $m^*$ . To argue indistinguishability in the adversary's view, we implicitly view  $r_{m_1,1} = \langle \text{ct}_{\sigma_{m_1}}, \text{fsk}_1 \rangle + r'_{m_1,1}$ , and write

$$\begin{aligned} \sigma_{m_1,2} &= \lfloor q/2 \rfloor \sigma_{m_1} - r_{m_1,1} + \text{noise} \\ &= \lfloor q/2 \rfloor \sigma_{m_1} - \langle \text{ct}_{\sigma_{m_1}}, \text{fsk}_1 \rangle - r'_{m_1,1} + \text{noise} \\ &= \lfloor q/2 \rfloor \sigma_{m_1} - \langle \text{ct}_{\sigma_{m_1}}, \text{fsk}_1 \rangle - \langle \text{ct}_{\sigma_{m_1}}, \text{fsk}_2 \rangle + \langle \text{ct}_{\sigma_{m_1}}, \text{fsk}_2 \rangle - r'_{m_1,1} + \text{noise} \\ &= \lfloor q/2 \rfloor \sigma_{m_1} - \langle \text{ct}_{\sigma_{m_1}}, \text{fsk} \rangle + \langle \text{ct}_{\sigma_{m_1}}, \text{fsk}_2 \rangle - r'_{m_1,1} + \text{noise} \\ &= \lfloor q/2 \rfloor \sigma_{m_1} - \lfloor q/2 \rfloor \sigma_{m_1} + e + \langle \text{ct}_{\sigma_{m_1}}, \text{fsk}_2 \rangle - r'_{m_1,1} + \text{noise} \\ &= \langle \text{ct}_{\sigma_{m_1}}, \text{fsk}_2 \rangle + (-r'_{m_1,1}) + e + \text{noise}. \end{aligned}$$

This view leaves the extra terms  $r'_{m_1,1}$  and  $(-r'_{m_1,1})$  in  $\sigma_{m_1,1}$  and  $\sigma_{m_1,2}$ , respectively. We can think of  $r'_{m_1,1}$  and  $(-r'_{m_1,1})$  as 2-out-of-2 random secret shares of 0. To address these extra random values, we modify the original construction by adding pseudorandom components to partial signatures as following. Let  $\text{PRF} : \mathcal{K} \times \{0, 1\}^\lambda \rightarrow \mathcal{Y}$  be a PRF. Then

$$\begin{aligned} \sigma_{m_1,1}^{\text{real}} &= \langle \text{ct}_{\sigma_{m_1}}, \text{fsk}_1 \rangle + \text{PRF}(K_1, m_1) + \text{noise}, \text{ and} \\ \sigma_{m_1,2}^{\text{real}} &= \langle \text{ct}_{\sigma_{m_1}}, \text{fsk}_2 \rangle + \text{PRF}(K_2, m_1) + \text{noise}. \end{aligned}$$

It is easy to see that for correctness,  $\text{PRF}(K_1, m_1) + \text{PRF}(K_2, m_1)$  must be zero<sup>5</sup>. However, for a general PRF for any two keys  $K$  and  $K'$ ,  $\text{PRF}(K, m_1) + \text{PRF}(K', m_1) \neq 0$  with high probability. So, we use a key homomorphic PRF and  $K_1, K_2$  are generated as 2-out-of-2 secret shares of  $0 \in \mathcal{K}$ . We recall that if PRF is key homomorphic, then for all  $K, K' \in \mathcal{K}$  and all input  $x$ ,  $\text{PRF}(K, x) + \text{PRF}(K', x) = \text{PRF}(K + K', x)$ . Thus, for  $K_2 = -K_1$ ,  $\text{PRF}(K_2, m_1) = \text{PRF}(-K_1, m_1) = -\text{PRF}(K_1, m_1)$ <sup>6</sup>. Now, we can argue indistinguishability of real partial signatures from simulated ones from PRF security that allows to replace  $\text{PRF}(K_1, m_1)$  with random  $r'_{m_1,1}$ , and also from the security of secret sharing that ensures that  $K_1$  is uniformly random.

In general, our modified construction is:

- $\text{sk}_i = (K_i, \text{fsk}_i)$ , where  $\{K_1, \dots, K_n\}$  are  $t$ -out-of- $n$  secret shares of  $0 \in \mathcal{K}$ .
- $\text{TS.PartSign}(i, \text{sk}_i, m)$  computes  $\sigma_{m,i} = \langle \text{ct}_{\sigma_m}, \text{fsk}_i \rangle + \text{PRF}(K_i, m) + \text{noise}_i$

A simple calculation shows the correctness. For security, we need that for  $\{K_1, \dots, K_n\}$  generated as secret shares of  $0 \in \mathcal{K}$  and for all input  $x \in \mathcal{X}$ , the following two distributions are indistinguishable (i)  $\{\text{PRF}(K_1, x), \dots, \text{PRF}(K_n, x)\}$  versus (ii)  $(r_{x,1}, \dots, r_{x,n})$ , where  $\{r_{x,1}, \dots, r_{x,n}\}$  are generated as secret shares of  $0 \in \mathcal{Y}$ . This should hold even if the adversary is given such samples for polynomially many different  $x$  of the adversary's choice. Here,  $\mathcal{K}$  is the key space,  $\mathcal{X}$ , the input space and  $\mathcal{Y}$  is the output space of the PRF. We prove this based on the security of PRF and the secret sharing scheme. Please see Sect. 4 for details.

*Lattice Based Key Homomorphic PRF?* Known lattice based constructions of key homomorphic PRF are only *almost* key homomorphic, that is  $\text{PRF}(K_1, x) + \text{PRF}(K_2, x) = \text{PRF}(K_1 + K_2, x) \pm \delta$ , where  $\delta$  is a small constant, mostly in  $\{0, 1, 2\}$ . Clearly, the above security property does not hold for almost key homomorphic PRF, because the adversary can combine the secrets using  $\text{SS.Combine}$  and if the result is zero, then it is the second case, else the first case, with good probability. We need extra care to work with almost key homomorphic PRF, where we also add some flooding noise to hide the error incurred by homomorphic evaluation of almost KHPRF. Please refer to Sect. 4 for details.

**Comparing with [ASY22].** In [ASY22], Agrawal *et. al* improve the BGGJKRS construction by providing adaptivity in the other direction as we discussed previously. They construct a threshold signature scheme that allows the adversary to output  $S^*$ , the set of parties for which it queries the signing keys, in the middle of the game, but does not allow partial signing queries on  $m^*$ . In that

<sup>5</sup> Indeed, it suffices that  $\text{PRF}(K_1, m_1) + \text{PRF}(K_2, m_1)$  can be computed publicly. In that case the  $\text{TS.Combine}$  algorithm can first compute and subtract the extra term  $\text{PRF}(K_1, m_1) + \text{PRF}(K_2, m_1)$  before rounding.

<sup>6</sup> For  $0 \in \mathcal{K}$ , any  $K \in \mathcal{K}$  and any input  $x$ ,  $\text{PRF}(K, x) = \text{PRF}(K + 0, x) = \text{PRF}(0, x) + \text{PRF}(K, x)$ . Hence,  $\text{PRF}(0, x) = 0$ . This further implies  $\text{PRF}(-K, x) = -\text{PRF}(K, x)$ .



case also, to answer the partial signature queries before key query, the challenger has similar dilemma - to decide which of the FHE key shares to use. However the nature of the problem is different - again considering the previous toy example of 2-out-of-2 access structure, suppose the challenger decides to use  $\text{fsk}_1$  and simulate the partial signatures from Party 2. That is, for any message  $m$ ,  $\sigma_{m,1} = \langle \text{ct}_{\sigma_m}, \text{fsk}_1 \rangle + \text{noise}$  and  $\sigma_{m,2} = \lfloor q/2 \rfloor \sigma_m - \langle \text{ct}_{\sigma_m}, \text{fsk}_1 \rangle + \text{noise}$ . This time there is no issue related to deciding whether to use  $\sigma_m$  or not. Instead, the issue is that if the adversary asks  $\text{fsk}_2$ , then the challenger will be in trouble, as it will end up using both the key shares of  $\text{fsk}$ . To address this situation, [ASY22] also use a similar idea - to simulate the partial signatures till  $S^*$  is received without using either  $\text{fsk}_1$  or  $\text{fsk}_2$ . That is, for a signing query on message  $m$ , the challenger computes  $\lfloor q/2 \rfloor \sigma_m$  and secret shares it as  $r_{m,1}$  and  $r_{m,2}$ . Then it sets  $\sigma_{m,1} = r_{m,1} + \text{noise}$  and  $\sigma_{m,2} = r_{m,2} + \text{noise}$ , and implicitly views  $r_{m,i} = \langle \text{ct}_{\sigma_m}, \text{fsk}_i \rangle + r'_{m,i}$ , for  $i = 1, 2$ . To account for extra  $r'_{m,1}$  and  $r'_{m,2}$ , Agrawal *et. al* also modify the BGGJKRS construction, but instead of PRF they use random oracle. In particular,  $\text{sk}_1 = (\text{fsk}_1, R_1, K)$ ,  $\text{sk}_2 = (\text{fsk}_2, R_2, K)$ . Here  $R_1$  and  $R_2$  are random vectors of length  $n$  (here 2) such that  $R_1 + R_2 = 0$  and  $H$  is a hash function modeled as random oracle.  $\text{PartSign}(\text{sk}_i, m) = \langle \text{ct}_{\sigma_m}, \text{fsk}_i \rangle + \langle H(K, m), R_i \rangle + \text{noise}$ , for  $i = 1, 2$ . When the adversary outputs  $S^*$ , the challenger does the following: for each message  $m$  for which a partial signature has been queried, it solves for  $h_m$  such that  $\langle h_m, R_i \rangle = r'_{m,i}$  for all  $i \in S^*$ . It then programs  $H(K, m) = h_m$  and returns  $\{\text{sk}_i\}_{i \in S^*}$ . In our toy example, let  $S^* = \{1\}$ , then the challenger solves for  $\langle h_m, R_1 \rangle = r'_{m,1}$ , programs  $H(K, m) = h_m$  and returns  $\text{sk}_1 = (\text{fsk}_1, R_1, K)$ . After this point, the challenger has no uncertainty and it behaves in the same way as in the proof of BGGJKRS. Observe that it is crucial that  $K$  has entropy and is hidden from the adversary until it outputs  $S^*$ . In fact, that is the reason this improvement gives only partial adaptivity and not the full adaptivity.

Looking back to our problem, we cannot use the ROM based trick from [ASY22] because, in our case, as soon as the adversary gets a signing key, it learns  $K$ , but the uncertainty for the challenger regarding when to use  $\text{fsk}_i$  or  $\sigma_m$  remains.

On the other hand, interestingly, the PRF based trick does not work for [ASY22]. This is because, if  $r'_{m,i}$  for  $i \in [n]$  replaces  $\text{PRF}(K_i, m)$ , then when the adversary outputs  $S^*$ , the challenger will need to provide a  $K'_i$  such that  $\text{PRF}(K'_i, m) = r'_{m,i}$ , which we don't know how to do. For us, the PRF based improvement works, because the challenger never needs to output such a key.

**Combining Our Approach with [ASY22] to Get Partially Adaptive Key Queries with Partial Signatures on  $m^*$ .** We can combine our PRF based technique with the ROM based technique in [ASY22] to get a construction that has the advantage of both the constructions. That is, the adversary can ask key queries in the middle of the game (but all at once) and also issue partial signing queries on  $m^*$ . We refer the readers to the full version<sup>7</sup> for more details.

<sup>7</sup> The full version is available on <https://eprint.iacr.org/>.

**Applying the Techniques at the TFHE Level.** In [BGG+18], Boneh *et. al* construct a threshold FHE (TFHE), using which they construct a universal thresholdizer (UT), which, in turn, is used to thresholdize various cryptographic primitives - threshold signature (TS), threshold CCA-PKE (TPKE).

We observed that all ideas and improvements that we discussed above in the context of threshold signatures, can in fact be implemented at the TFHE level itself, which then will have the obvious advantage of improving the security of all the threshold primitives that are thresholdized from TFHE.

A TFHE is the same as FHE except that the decryption algorithm is thresholdized in TFHE. In more detail,

- **Setup** algorithm outputs a public key  $\text{fpk}$  and  $n$  secret keys  $\{\text{fsk}_i\}_{i \in [n]}$ .
- The **Eval** algorithm is the same as the one in FHE. It takes as input, the ciphertexts  $\text{ct}_1, \dots, \text{ct}_k$ , where  $\text{ct}_i = \text{Encrypt}(\mu_i)$ ,  $\mu_i \in \{0, 1\}$ , and a circuit  $C : \{0, 1\}^k \rightarrow \{0, 1\}$  and outputs a ciphertext  $\text{ct}$  that encrypts  $C(\mu_1, \dots, \mu_k)$ .
- The **PartDec** algorithm takes as input a partial decryption key  $\text{fsk}_i$  and a ciphertext  $\text{ct}$  and outputs a partial decryption  $p_i$ .
- The **FinDec** algorithm takes as input a set of partial decryptions of a ciphertext from a valid set of parties (i.e. the number of parties in the set is at least  $t$ ) and outputs the encrypted message.

Boneh *et. al* define semantic security and simulation security for TFHE. The semantic security is the same as the semantic security of any PKE. Roughly speaking simulation security is defined as follows. There exists an efficient simulator  $\mathcal{S}$ , which can simulate the secret key shares  $\{\text{fsk}_i\}_{i \in [n]}$ , without using  $\text{fsk}$  in the **Setup**. Later on, given a set of ciphertexts  $\{\text{ct}_1, \dots, \text{ct}_k\}$  which encrypts adversarially chosen message bits  $\{\mu_1, \dots, \mu_k\}$ , and any circuit  $C : \{0, 1\}^k \rightarrow \{0, 1\}$  along with  $C(\mu_1, \dots, \mu_k)$  and a set  $S$ ,  $\mathcal{S}$  simulates partial decryptions of  $\text{ct}_C$  on behalf of parties in the set  $S$ . Here  $\text{ct}_C = \text{TFHE.Eval}(\{\text{ct}_1, \dots, \text{ct}_k\}, C)$ . The simulator does not need to know the message bits  $\mu_1, \dots, \mu_k$ . The simulation security says that for any PPT adversary who gets partial decryption keys  $\{\text{fsk}_i\}_{i \in S^*}$  corresponding to any invalid set of parties of its choice in the beginning, the simulated view is indistinguishable from the real view. For precise definition, we refer to the full version. For the current discussion, we only need to observe that in this definition, the simulator  $\mathcal{S}$  takes  $C(\mu_1, \dots, \mu_k)$  as input irrespective of whether  $S$  is a valid (i.e.  $|S| \geq t$ ) or an invalid set ( $|S| < t$ ). We observed that this is the main hurdle in proving stronger security of threshold signature (and also the other primitives) built from the universal thresholdizer (UT) in [BGG+18]. We propose stronger definition of simulation security for TFHE, where the simulator  $\mathcal{S}$  needs  $C(\mu_1, \dots, \mu_k)$  only when it generates partial decryption for a valid set of parties. We then propose the same improvement in UT security, which in turn allows us to prove the stronger security for thresholdized primitives. We apply the ideas discussed above in the context of TSig at the level of TFHE to achieve the stronger simulation security.

**Stronger Security for Threshold CCA-PKE.** We define a stronger CCA security definition for a threshold public-key encryption scheme. In a real life attack

scenario, an adversary capable of corrupting a number of parties below the threshold value can participate in the decryption of a targeted ciphertext  $\text{ct}^*$  with the help of a corrupt party. Let  $g = t - |S^*|$  be the gap between the threshold value and the number of corrupt parties. A natural notion of CCA security should allow up to  $g - 1$  number of partial decryption queries on the challenge ciphertext  $\text{ct}^*$ . We define a CCA security definition in which the adversary is allowed to query partial decryption queries on  $\text{ct}^*$  from up to  $g - 1$  honest parties. Furthermore, we show that the threshold public-key encryption scheme in [BGG+18] constructed from a UT scheme and a public-key encryption satisfies our stronger notion of CCA security if UT is strongly secure (which we define) and the underlying public-key encryption is CCA secure.

**Other Related Work.** Significant research has been done in building lattice based threshold cryptography with efficient parameters. Unfortunately efficient constructions are achieved at the cost of increasing the number of communication rounds between the parties [dPKM+24, GKS24] or restricting to  $n$ -out-of- $n$  access structure [DOTT21, MS23]. In [dPKM+24] del Pino *et. al* construct an efficient threshold signature that involves 3 rounds. In [GKS24], Gur, Katz and Silde improve [ASY22] but at the cost of increasing the number of rounds to 2. Moreover both the constructions achieve weaker notion of unforgeability ( $P_1$  in Fig. 1). In [CCK23] improve the efficiency of [BGG+18] UT using iterative Shamir secret sharing. However, they consider the same (weaker) security definition for UT as in [BGG+18].

*Organization of the Paper:* We provide the notations and preliminaries in Sects. 2. For detailed preliminaries, please see the full version. In Sect. 3 we define simulators for secret sharing which we use in our construction. In Sect. 4, we present key homomorphic PRF. In Sect. 5, we define stronger definition of TFHE and provide our construction that satisfies the stronger security. In Sect. 6, we define and prove stronger security definition of UT. In Sect. 7, we prove stronger security of threshold signatures and threshold PKE from strongly secure UT.

## 2 Preliminaries

*Notations:* We represent the  $t$ -out-of- $n$  threshold structure as  $\mathbb{A}_{t,n}$ . At many places we refer to a party  $P_i$  as  $i$  only (i.e. discarding the letter  $P$ ). For secret sharing scheme where each party's shares consist of multiple shares, we refer to each share index by the term 'party-share' or only 'share'. For a matrix  $\mathbf{M}$  of dimensions  $\ell \times N$ , we represent the  $i$ -th row as  $\mathbf{M}[i]$  or  $\mathbf{M}_i$ . Let  $S \subseteq [\ell]$ , then we use  $\mathbf{M}[S]$  or  $\mathbf{M}_S$  to represent the matrix formed by rows in set  $S$ . For a vector  $\mathbf{x}$ , unless otherwise stated,  $x_i$  represents the  $i$ -th element in  $\mathbf{x}$ . Due to space constraints we provide necessary preliminaries in the full version.

### 3 Secret Sharing

We first recall the definitions related to secret sharing. Then we define two simulators that can simulate secret shares for parties one by one as per the need. More importantly, the simulators do not need the actual secret till the shares are generated for less than the threshold number of parties. Here we recall the definition of linear secret sharing. For other preliminaries related to access structures and secret sharing, please refer to the full version.

**Definition 1 (Linear Secret Sharing (LSSS)).** *Let  $P = \{P_i\}_{i \in [n]}$  be a set of parties and  $\mathbb{S}$  be a class of efficient access structures. A secret sharing scheme  $\text{SS}$  with secret space  $\mathcal{K} = \mathbb{Z}_p$  for some prime  $p$  is called a linear secret sharing scheme if it satisfies the following properties:*

- $\text{SS.Share}(\mathbf{k}, \mathbb{A})$ : *There exists a matrix  $\mathbf{M} \in \mathbb{Z}_p^{\ell \times N}$  called the share matrix, and each party  $P_i$  is associated with a partition  $T_i \subseteq [\ell]$ . We assume a  $\text{SS.Setup}$  algorithm that outputs the share matrix and the partitions as  $\text{pp}$  which is implicitly assumed to be an input to the  $\text{SS.Share}$  and the  $\text{SS.Combine}$  algorithm.*

*To create the shares on a secret  $\mathbf{k}$ , the sharing algorithm first samples uniform values  $r_1, \dots, r_{N-1} \leftarrow \mathbb{Z}_p$  and defines a vector  $\mathbf{w} = \mathbf{M} \cdot (\mathbf{k}, r_1, \dots, r_{N-1})^\top$ . The share  $k_i$  for  $P_i$  consists of the entries  $k_i = \{w_j\}_{j \in T_i}$ .*

*At many places in the paper, when it is clear from the context, we directly write  $(w_1, \dots, w_\ell) \leftarrow \text{SS.Share}(\mathbf{k}, \mathbb{A})$  instead of  $(k_1, \dots, k_n) \leftarrow \text{SS.Share}(\mathbf{k}, \mathbb{A})$  with  $k_i = \{w_j\}_{j \in T_i}$ .*

- $\text{SS.Combine}(B)$ : *For any valid set  $S \in \mathbb{A}$ , we have*

$$(1, 0, \dots, 0) \in \text{span}(\{\mathbf{M}[j]\}_{j \in \bigcup_{i \in S} T_i}).$$

*over  $\mathbb{Z}_p$  where  $\mathbf{M}[j]$  denotes the  $j$ th row of  $\mathbf{M}$ . Any valid set of parties  $S \in \mathbb{A}$  can efficiently find the coefficients  $\{c_j\}_{j \in \bigcup_{i \in S} T_i}$  satisfying*

$$\sum_{j \in \bigcup_{i \in S} T_i} c_j \cdot \mathbf{M}[j] = (1, 0, \dots, 0)$$

*and recover the secret by computing  $\mathbf{k} = \sum_{j \in \bigcup_{i \in S} T_i} c_j \cdot w_j$ . The coefficients  $\{c_j\}$  are called recovery coefficients.*

To secret-share a vector  $\mathbf{s} = \{s_1, \dots, s_m\} \in \mathbb{Z}_p^m$ , we can simply secret-share each entry  $s_i$  using fresh randomness. This gives secret share vectors  $\mathbf{s}_1, \dots, \mathbf{s}_\ell \in \mathbb{Z}_p^m$ . Using these secret shares, the secret vector  $\mathbf{s}$  can be recovered using the same coefficients as that for a single field element.

Below we describe the properties of two kinds of linear secret sharing schemes for threshold access structure (TAS) - (i) Shamir secret sharing and (ii)  $\{0, 1\}$ -linear secret sharing.

**Theorem 1 (Shamir Secret Sharing).** *Let  $P = \{P_1, \dots, P_n\}$  be a set of parties, and let TAS be the class of threshold access structures on  $P$ . Then, there exists a linear secret sharing scheme (Definition 1) SS with secret space  $\mathcal{K} = \mathbb{Z}_p$  for some prime  $p$  satisfying the following properties:*

- *For any secret  $s \in \mathbb{Z}_p$  and  $\mathbb{A}_{t,n} \in \text{TAS}$ , each share for party  $P_i$  consists of a single element  $w_i \in \mathbb{Z}_p$ . Let us denote  $w_0 = s$ .*
- *For every  $i, j \in [n] \cup \{0\}$  and set  $S \subset [n] \cup \{0\}$  of size  $t$ , there exists an efficiently computable Lagrange coefficients  $\gamma_{i,j}^S \in \mathbb{Z}_p$  such that  $w_j = \sum_{i \in S} \gamma_{i,j}^S \cdot w_i$ .*

The following lemma can be stated about TAS from [BGG+18].

**Lemma 1 ( $\{0, 1\}$ -LSSS for TAS [BGG+18]).** *Let  $P = \{P_1, \dots, P_n\}$  be a set of parties. Let  $\mathbb{A}_{t,n}$  be a  $t$ -out-of- $n$  threshold access structure. There exists an efficient linear secret sharing scheme  $\text{bSS} = (\text{bSS.Share}, \text{bSS.Combine})$  over the secret space  $\mathcal{K} = \mathbb{Z}_p$  satisfying the following property:*

- *Let  $s$  be a shared secret and  $\{w_j\}_{j \in T_i}$  be the share of party  $P_i$  for  $i \in [n]$ . Then, for every set  $S \in \mathbb{A}_{t,n}$ , there exists a subset  $T \subseteq \bigcup_{i \in S} T_i$  such that  $s = \sum_{j \in T} w_j$ . Moreover the set  $T$  can be computed efficiently for all  $S \in \mathbb{A}_{t,n}$ .*

We call a linear secret sharing scheme that satisfy the properties above as a  $\{0, 1\}$ -linear secret sharing scheme<sup>8</sup>.

Note that for any  $\{0, 1\}$ -linear secret sharing scheme  $\text{bSS}$ , and for any minimal valid share set  $T \subseteq [\ell]$ , we have that  $\sum_{j \in T} w_j = s$ . We will use  $\text{bSS} = (\text{bSS.Share}, \text{bSS.Combine})$  to refer to a  $\{0, 1\}$ -LSSS in this work.

### 3.1 LSSS Simulators

In the above descriptions, the  $\text{SS.Share}$  algorithm takes the secret as input and outputs the secret shares for all the parties in one step. However, the secret shares can indeed be generated in a sequential manner - one party (or even one share) at a time - and the secret is needed only when the set of parties (or shares) become valid. This is straightforward, for example, for  $n$ -out-of- $n$  secret sharing, where  $n - 1$  secret shares are chosen independently randomly as  $r_1, \dots, r_{n-1}$  and then the last share is computed as  $s - \sum_{i \in [n-1]} r_i$ . However, for more general case of  $t$ -out-of- $n$  sharing, where each party can have more than one secret share, it is little more involved. We formalize this by defining two simulators -  $\text{SSSiml}$  and  $\text{SSSimV}$  as follows:

- $\text{SSSiml}(\text{pp}, S, \{w_i\}_{i \in S}, R, \text{st}) \rightarrow (\{w_j\}_{j \in R}, \text{st}')$ : takes as input a set  $S \subseteq [\ell]$ , for which the secret shares are already assigned, along with the assigned secret shares  $\{w_i\}_{i \in S}$ , a subset  $R \subseteq [\ell]$ , for which the secret shares are to be

---

<sup>8</sup> We are using a slightly different naming from [BGG+18]. They call such a scheme a *special* linear secret sharing scheme and the class of access structures that supports special LSS as  $\{0, 1\}$ -LSSS.

generated and a state  $\mathbf{st}$ , and outputs the secret shares for indices in  $R$  and a new state  $\mathbf{st}'$ . Wlog, we assume  $R \cap S = \emptyset$ , otherwise for such  $j \in R \cap S$ , the simulator simply returns the  $w_j$  from the input secret shares and works with  $R \setminus S$ . We note that the simulator does not take the secret  $s$  being shared and that  $S \cup R$  must be an invalid share set.

1. If  $S \cup R$  is a valid share set, then return  $\perp$ .
  2. Initialize  $I = \emptyset$  and  $W = S$ .
  3. For each  $j \in R$ ,
    - If  $\mathbf{M}[j] \notin \text{Span}(\mathbf{M}_W)$  (i.e., is independent of rows in  $\mathbf{M}_W$ ), then  $I = I \cup \{j\}$  and  $W = W \cup \{j\}$ ; and  $w_j \leftarrow \mathbb{Z}_q$ .
    - Else,  $\exists \{\gamma_\alpha\}_{\alpha \in W}$  such that  $\mathbf{M}[j] = \sum_{\alpha \in W} \gamma_\alpha \mathbf{M}[\alpha]$ . Set  $w_j = \sum_{\alpha \in W} \gamma_\alpha w_\alpha$ .
  4. Return  $\{w_\alpha\}_{\alpha \in R}$  and  $\mathbf{st}' = \mathbf{st}$ .
- $\text{SSSimV}(\text{pp}, S, \{w_i\}_{i \in S}, s, R, \mathbf{st}) \rightarrow (\{w_j\}_{j \in R}, \mathbf{st}')$ : takes as input a set  $S \subseteq [\ell]$ , for which the secret shares are already assigned, along with the assigned secret shares  $\{w_i\}_{i \in S}$ , the secret  $s$ , a subset  $R \subseteq [\ell]$ , for which the secret shares are to be generated and a state  $\mathbf{st}$ , and outputs the secret shares for indices in  $R$  and a new state  $\mathbf{st}'$ . Wlog, we assume  $R \cap S = \emptyset$ , otherwise for such  $j \in R \cap S$ , the simulator simply returns the  $w_j$  from the input secret shares and work with  $R \setminus S$ . We note that this simulator takes the secret  $s$  being shared and that  $S \cup R$  is a valid share set.
1. If  $\mathbf{st} = \emptyset$ 
    - (a) If  $S$  is a valid share set, then return  $\perp$ .
    - (b) Find a maximal invalid share set  $X \subseteq [\ell]$  such that  $S \subseteq X$ .
    - (c) Initialize  $I = \emptyset$  and  $W = S$ .
    - (d) For  $j \in X \setminus S$ ,
      - i. If  $\mathbf{M}[j] \notin \text{Span}(\mathbf{M}_W)$ , sample  $w_j \leftarrow \mathbb{Z}_q$  and update  $I = I \cup \{j\}$  and  $W = W \cup \{j\}$ .
      - ii. Else compute  $w_j$  from  $\{w_\alpha\}_{\alpha \in W}$  in the same ways as in Step 3 in  $\text{SSSim1}$  above.
    - (e)  $\mathbf{st}' = (X, \{w_\alpha\}_{\alpha \in X})$
    - (f) For  $j \in R \setminus X$ 
      - i. Since  $X$  is a *maximally* invalid share set,  $X \cup \{j\}$  is a valid share set. Hence, there exists  $\{c_\alpha\}_{\alpha \in X \cup \{j\}}$  such that  $s = \sum_{\alpha \in X \cup \{j\}} c_\alpha w_\alpha$ . Hence, compute  $w_j = \frac{s - \sum_{\alpha \in X} c_\alpha w_\alpha}{c_j}$ .
    - (g) Return  $\{w_\alpha\}_{\alpha \in R}$  and  $\mathbf{st}'$ .
  2. Else
    - (a) Parse  $\mathbf{st}$  as  $(X, \{w_\alpha\}_{\alpha \in X})$  where  $X$  is a maximally invalid share set - if not, abort.
    - (b) Again, wlog we assume  $R \cap X = \emptyset$ , because otherwise for all  $j \in X \setminus R$ ,  $w_j$  is already set and is returned as it is.
    - (c) For  $j \in R \setminus X$

- i. Since  $X$  is a *maximally* invalid share set,  $X \cup \{j\}$  is a valid share set. Hence, there exists  $\{c_\alpha^{X \cup \{j\}}\}_{\alpha \in X \cup \{j\}}$  such that  $s = \sum_{\alpha \in X \cup \{j\}} c_\alpha^{X \cup \{j\}} w_\alpha$ . Hence, compute  $w_j = \frac{s - \sum_{\alpha \in X} c_\alpha^{X \cup \{j\}} w_\alpha}{c_j^{X \cup \{j\}}}$
- 3. Return  $\{w_\alpha\}_{\alpha \in R}$  and  $\text{st}' = \text{st}$ .

Below we show that the secret shares generated by the simulators  $\text{SSSimI}$  and  $\text{SSSimV}$  have the same distribution as those generated by the  $\text{SS.Share}$  algorithm. We formalize this via the following claim.

**Claim 2.** *For all adversary  $\mathcal{A}$ ,  $\text{Expt}_{\text{SSSim}}^0$  and  $\text{Expt}_{\text{SSSim}}^1$ , as described below are identical in  $\mathcal{A}$ 's view.*

$\text{Expt}_{\text{SSSim}}^b(1^\lambda)$  :

1. Upon input the security parameter  $1^\lambda$  and a threshold access structure  $\mathbb{A}_{t,n}$ , the challenger fixes a share matrix  $\mathbf{M}$  and the partitions  $\{T_i\}_{i \in [n]}$  and sends them to  $\mathcal{A}$ .
2.  $\mathcal{A}$  outputs a secret  $s$  (for which secret shares are to be generated).
3. The challenger does the following:
  - If  $b = 0$ , generates  $\{w_j\}_{j \in [\ell]} \leftarrow \text{SS.Share}(s, \mathbb{A}_{t,n})$
  - Else if  $b = 1$ , initializes  $W = \emptyset$  and  $\text{st} = \emptyset$ .
4. For  $\kappa = 1$  to  $\ell$ ,
  - (a)  $\mathcal{A}$  outputs  $j_\kappa \in [\ell]$  (wlog we assume that  $j_\kappa$  was not queried before).
  - (b) If  $b = 0$ , the challenger returns  $w_{j_\kappa}$  (generated in Step 3)
  - (c) If  $b = 1$  and  $W \cup \{j_\kappa\}$  is an invalid share set, generate  $(w_{j_\kappa}, \text{st}') \leftarrow \text{SSSimI}(W, \{w_\alpha\}_{\alpha \in W}, \{j_\kappa\}, \text{st})$ ; updates  $W = W \cup \{j_\kappa\}, \text{st} = \text{st}'$  and returns  $w_{j_\kappa}$
  - (d) If  $b = 1$  and  $W \cup \{j_\kappa\}$  is a valid share set, generate  $(w_{j_\kappa}, \text{st}') \leftarrow \text{SSSimV}(W, \{w_\alpha\}_{\alpha \in W}, s, \{j_\kappa\}, \text{st})$ ; updates  $W = W \cup \{j_\kappa\}, \text{st} = \text{st}'$  and returns  $w_{j_\kappa}$
5. At the end,  $\mathcal{A}$  outputs its guess bit  $b'$  and wins if  $b' = b$ .

*Proof.* We first recall the  $\text{SS.Share}$  algorithm. The secret shares of  $s$ ,  $\{w_j\}_{j \in [\ell]}$  are computed as  $\mathbf{M} \cdot (s, r_1, \dots, r_{N-1})^\top = (w_1, \dots, w_\ell)^\top$ , where  $r_1, \dots, r_{N-1}$  are chosen uniformly randomly.

To prove the claim, we show that the secret shares  $\{w_j\}_{j \in [\ell]}$  generated by the simulators can also be expressed as  $\mathbf{M} \cdot (s, r_1, \dots, r_{N-1})^\top = (w_1, \dots, w_\ell)^\top$ , for uniformly random  $r_1, \dots, r_{N-1}$ .

Let  $k \in [\ell]$  be the index where the transition from invalid to valid happens - i.e.,  $\{j_1, \dots, j_k\}$  is an invalid share set and  $\{j_1, \dots, j_k, j_{k+1}\}$  is a valid share set. Let  $I$  be the maximal invalid share set chosen by  $\text{SSSimV}$  when called to generate  $w_{j_{k+1}}$ . Note that for all subsequent queries also, the same invalid share set  $I$  is used. Further, let  $E \subseteq I$  be the set of indices for which the secret shares are chosen uniformly randomly. From the definition of the simulators, we note that the rows in  $\mathbf{M}_E$  are independent. Let  $|E| = c$ . Now we make the following observations: since  $I$  is a maximally invalid share set and  $E \subseteq I$ ,  $E$  is also an invalid share set. Hence, for all  $j \in [\ell] \setminus I$ ,  $\mathbf{M}[j] \notin \text{Span}(\mathbf{M}_E)$ , otherwise the

adversary could recover the shares for a valid share set  $I \cup \{j\}$  from the shares of an invalid share set  $I$ , where the  $j$ -th share  $w_j$  could be generated from  $\{w_\alpha\}_{\alpha \in E}$ , thus breaking the security of secret sharing scheme. Thus,  $\text{rank}(\mathbf{M}) \geq |E| + 1$  and hence,  $N \geq |E| + 1$  or  $|E| \leq N - 1$ .

Case 1:  $|E| = N - 1$ , then given  $\mathbf{M}_E$  and  $\{w_j\}_{j \in E}$ , we can uniquely solve for  $r_1, \dots, r_{N-1}$  such that  $\mathbf{M}_E \cdot (s, r_1, \dots, r_{N-1})^\top = (w_j)_{j \in E}$ , and since  $\{w_j\}_{j \in E}$  are also chosen uniformly randomly from the same space as  $r$ 's,  $\{r_j\}_{j \in [N-1]}$  are also uniformly random.

Case 2:  $|E| < N - 1$ . choose  $r_1, \dots, r_{N-1-|E|}$  uniformly randomly and then uniquely solve for the rest of  $r$ 's, i.e.  $r_{N-1-|E|}$  to  $r_{N-1}$ , as in case 1, and by the same argument, these  $r$ 's are also uniformly random.

Finally, it is straightforward to verify that for any  $(s, r_1, \dots, r_{N-1})$ , such that  $\forall j \in E, \mathbf{M}[j] \cdot (s, r_1, \dots, r_{N-1})^\top = w_j, \mathbf{M}[\kappa] \cdot (s, r_1, \dots, r_{N-1})^\top = w_\kappa$ , for all  $\kappa \in [\ell] \setminus E$ .

## 4 Key Homomorphic PRF

We recall the definitions related to pseudorandom functions.

**Definition 2 (Pseudorandom Function (PRF)).** A function  $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$  with key space  $\mathcal{K}$ , domain  $\mathcal{X}$ , and range  $\mathcal{Y}$  is a secure pseudorandom function if for all efficient algorithms  $\mathcal{A}$ ,

$$|\Pr[k \leftarrow \mathcal{K} : \mathcal{A}^{F(k, \cdot)}(1^\lambda) = 1] - \Pr[f \leftarrow \text{Funcs}(\mathcal{X}, \mathcal{Y}) : \mathcal{A}^{f(\cdot)}(1^\lambda) = 1]| = \text{neg}(\lambda),$$

where  $\text{Funcs}(\mathcal{X}, \mathcal{Y})$  denotes the set of all functions with domain  $\mathcal{X}$  and range  $\mathcal{Y}$ .

**Definition 3 (Key Homomorphic PRF).** Any function  $F : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{Y}$  is a key homomorphic PRF (KHPRF) [NPR99, BLMR13] if it satisfies the following two properties:

1. It must be a PRF.
2. It satisfies key homomorphism: for any  $k_1, k_2 \in \mathcal{K}$ ,  $F(k_1, x) + F(k_2, x) = F(k_1 + k_2, x)$  for all  $x \in \mathcal{X}$ <sup>9</sup>.

**Definition 4 (Almost Key Homomorphic PRF).** A  $\delta$ -almost KHPRF [BLMR13] is the same as the standard PRF except that the second condition is different as:

1. It satisfies (almost) key homomorphism: for any  $k_1, k_2 \in \mathcal{K}$ ,  $F(k_1, x) + F(k_2, x) = F(k_1 + k_2, x) + e$ , where  $|e| \leq \delta$ , for all  $x \in \{0, 1\}^a$ .

<sup>9</sup> In general, if  $\mathcal{K}$  is a group with operation '+' and  $\mathcal{Y}$  is a group with operation '\*', then  $F(k_1, x) * F(k_2, x) = F(k_1 + k_2, x)$ .



Almost KHPRF are constructed from LWE in [BLMR13, BP14, Kim20], where  $\delta = 1$  or 2, depending upon the choice of the parameters.

We prove the following lemma which says that: for a secure KHPRF  $F$  and linear secret sharing scheme  $\text{SS}$ , let  $K$  is a KHPRF key and is secret shared as  $(K_1, \dots, K_n) \leftarrow \text{SS.Share}(K, \mathbb{A}_{t,n})$ . Then for all PPT adversary  $\mathcal{A}$ , who outputs polynomially many queries of the form  $(i, x)$ , the following two views are indistinguishable - in the first (real) world,  $\mathcal{A}$  receives  $F(K_i, x)$ , while in the other (ideal) world,  $\mathcal{A}$  receives  $R_{x,i}$ , where  $(R_{x,1}, \dots, R_{x,n}) \leftarrow \text{SS.Share}(F(K, x), \mathbb{A}_{t,n})$ <sup>10</sup>. This is true even if the adversary can corrupt up to  $t - 1$  parties (now outputs for only uncorrupted parties are random in the ideal world) and also knows  $K$ . Intuitively, this holds because, from the security of  $\text{SS}$ , the key shares  $K_1, \dots, K_n$  are “random” with the constraint that any valid combination of these keys gives  $K$ . Hence, from KHPRF security we can replace  $F(K_i, x)$  in the real world with random  $r_{x,i}$  in the ideal world under the constraint that any valid combination of  $r_{x,\cdot}$  gives  $F(K, x)$ . These  $\{r_{x,i}\}_{i \in [n]}$  can indeed be generated as secret shares of  $F(K, x)$ .

However, observe that in the case of almost KHPRF, the adversary can distinguish between the two worlds as follows. Let us consider a simple case of 2-out-of-2 sharing of an almost KHPRF key  $k$  (known to the adversary) as  $k = k_1 + k_2$  without any corruption. On any input  $x$ , the adversary is given  $F(k_1, x)$  and  $F(k_2, x)$  in the real world. In the ideal world, the adversary is given a random secret shares of  $F(k, x) : r_1, r_2$  such that  $r_1 + r_2 = F(k, x)$ . The adversary can distinguish the two worlds by adding the received values - if they add up exactly to  $F(k, x)$ , then it is the ideal world, else the real world with “good” probability. Hence, in almost KHPRF, we must add noise to hide the error introduced due to homomorphic evaluation.

We further observe that since each addition may add a  $\delta$  error, the total error introduced due to the homomorphic evaluation of partially evaluated PRF values (let us call it  $e_p$ ) may actually depend on the secret sharing schemes - in particular, the recovery coefficients. In  $\{0, 1\}$ -LSSS, the recovery co-efficients are binary and hence it is easier to bound the error as  $|e_p| \leq \ell$ , where  $\ell$  is the number of rows in the share matrix  $\mathbf{M}$ . In Shamir secret sharing the recovery co-efficients can be arbitrary in  $\mathbb{Z}_p$ . Hence, in that case one would need the technique of ‘clearing the denominators’ as in [ABV+12, BGG+18] and modify the game accordingly. In this paper, we work with  $\{0, 1\}$ -LSSS. However, the same ideas work for Shamir secret sharing as well.

Below we state and prove the lemma directly for the general case of almost KHPRF for  $\{0, 1\}$ -LSSS.

Before formally defining the lemma, let us define an intermediate algorithm  $\text{SS.ShareInt}$  for generating secret shares when some of the secret shares are already set. Thus,  $\text{SS.ShareInt}(\text{pp}, S, \{w_j\}_{j \in S}, s)$  takes as input the public parameters, an invalid set of party shares  $S \subset [\ell]$  and the corresponding secret shares,  $\{w_j\}_{j \in S}$  and the secret  $s$  and outputs the secret shares for  $[\ell] \setminus S$ . We assume

<sup>10</sup> Each  $K_i$  may indeed consist of multiple keys as  $K_i = \{k_j\}_{j \in T_i}$ . In that case,  $F(K_i, x) = \{F(k_j, x)\}_{j \in T_i}$  and  $R_{x,i} = \{r_{x,j}\}_{j \in T_i}$ .

that the input shares are consistent in the following sense: for all  $j \in S$  such that the row  $\mathbf{M}[j] \in \text{Span}(\mathbf{M}_{S \setminus \{j\}})$ , that is,  $\mathbf{M}[j] = \sum_{\kappa \in S \setminus \{j\}} c_\kappa \mathbf{M}[\kappa]$ , where  $\{c_\kappa\}_{\kappa \in S \setminus \{j\}}$  are constants,  $w_j = \sum_{\kappa \in S \setminus \{j\}} c_\kappa w_\kappa$ . The algorithm is defined as follows:

SS.ShareInt( $S, \{w_j\}_{j \in S}, s, \mathbb{A}_{t,n}$ ):

- Find a set of *random* values  $r_1, \dots, r_{N-1}$  such that for each  $\alpha \in S$ ,  $\langle \mathbf{M}[\alpha], (s, r_1, \dots, r_{N-1}) \rangle = w_\alpha$ . These  $r$ 's are chosen as follows:
  - Let  $S_I \subseteq S$  such that  $\mathbf{M}_{S_I}$  is maximally independent set of rows within  $\mathbf{M}_S$ . Let  $|S_I| = c$ .
  - Sample  $r_1, \dots, r_{N-1-c} \leftarrow \mathbb{Z}_q$ .
  - Solve (deterministically)  $\mathbf{M}_{S_I}(s, r_1, \dots, r_{N-1-c}, r_{N-1-c+1}, \dots, r_{N-1})^\top = w_{S_I}$  to compute  $r_{N-1-c+1}, \dots, r_{N-1}$ .
  - For each  $\alpha \in [\ell] \setminus S$ ,  $w_\alpha = \langle \mathbf{M}_\alpha, (s, r_1, \dots, r_{N-1}) \rangle$

Now we are ready to define the lemma.

**Lemma 2.** *Let  $F$  be any secure  $\delta$ -almost KHPRF and SS is a secure linear secret sharing scheme (Definition 1). Then for all PPT adversary  $\mathcal{A}$ ,  $\Pr[\mathcal{A} \text{ wins}] \leq 1/2 + \text{neg}(\lambda)$  in the following experiments if  $\delta\ell/E_{sm} \leq \text{neg}(\lambda)$ , where  $E_{sm}$  is flooding noise used to hide the error in homomorphic evaluation of KHPRF.*

Expt $_{\mathcal{A}, \text{almost-KHPRF}}(1^\lambda, \mathbb{A}_{t,n})$ :

1. Upon input the security parameter  $\lambda$  and a threshold access structure  $\mathbb{A}_{t,n}$ , the challenger  $\mathcal{C}$  finds a share matrix  $\mathbf{M}$  of dimensions  $\ell \times N$  along with the  $n$  partitions as  $(\mathbf{M}, \{T_i\}_{i \in [n]}) \leftarrow \text{bSS.Setup}(1^\lambda, \mathbb{A}_{t,n})$ . It sends  $\text{bSS.pp} = (\mathbf{M}, \{T_i\}_{i \in [n]})$  to  $\mathcal{A}$ .
2.  $\mathcal{C}$  samples a challenge bit,  $b \leftarrow \{0, 1\}$ .
3.  $\mathcal{A}$  outputs a PRF key  $K$  and an invalid share set  $S^* \subseteq [\ell]$ .
4.  $\mathcal{C}$  runs  $\{k_1, \dots, k_\ell\} \leftarrow \text{bSS.Share}(K, \mathbb{A}_{t,n})$  and returns  $\{k_j\}_{j \in S^*}$  to  $\mathcal{A}$ .
5. Then  $\mathcal{A}$  issues polynomial number of evaluation queries of the form  $(x, j)$  adaptively, where  $j \in [\ell]$  and  $x$  is an input to PRF  $F$ .
6. For each evaluation query  $(x, j)$ ,  $\mathcal{C}$  does the following.
  - Samples  $\eta_{x,j} \leftarrow [-E_{sm}, E_{sm}]$ , where  $\delta\ell/E_{sm} \leq \text{neg}(\lambda)$ .
  - If  $b = 0$ , it returns  $y_{x,j} = F(k_j, x) + \eta_{x,j}$ .
  - Else,
    - if  $j \in S^*$ , return  $y_{x,j} = F(k_j, x) + \eta_{x,j}$ .
    - else,  $\mathcal{C}$  does the following:
      - if  $x$  is queried for the first time (irrespective of any  $j \in [\ell]$ ), then it first computes  $y_{x,\alpha} = F(k_\alpha, x)$  for all  $\alpha \in S^*$  and runs  $\{y_{x,\alpha}\}_{\alpha \in [\ell] \setminus S^*} \leftarrow \text{bSS.ShareInt}(\text{SS.pp}, S^*, \{y_{x,\alpha}\}_{\alpha \in S^*}, F(K, x), \mathbb{A}_{t,n})$  and returns  $y_{x,j} + \eta_{x,j}$ . It saves  $\{y_{x,\alpha}\}_{\alpha \in [\ell] \setminus S^*}$  for future queries.
      - else  $\mathcal{C}$  returns the saved  $y_{x,j} + \eta_{x,j}$ .
7.  $\mathcal{A}$  outputs  $b'$  and wins if  $b' = b$ .

- Here, we are using a slight abuse of notation - we work directly at the level of party shares in  $[\ell]$  without identifying the parties to which they belong. For example, we write  $(k_1, \dots, k_\ell) \leftarrow \text{bSS.Share}(K, \mathbb{A}_{t,n})$  instead of  $\{\{k_j\}_{j \in T_i}\}_{i \in [n]} \leftarrow \text{bSS.Share}(K, \mathbb{A}_{t,n})$ .
- In the above game, the adversary can issue evaluation queries for multiple party shares together as  $(S, x)$ , where  $S \subseteq [\ell]$ .

Due to space constraints, we prove the lemma in the full version.

## 5 Threshold Fully Homomorphic Encryption

We recall the definitions of TFHE from [BGG+18] along with its correctness and security (semantic and simulation) properties in the full version of the paper. Here we define our stronger notion of simulation security needed for constructing UT with stronger security, which in turn is needed for building thresholdized primitives with stronger notion of security.

**Definition 5 (Stronger Simulation Security).** *A TFHE scheme satisfies stronger simulation security if for all  $\lambda$ , depth bound  $d$ , and access structure  $\mathbb{A}$ , the following holds. There exists a stateful PPT algorithm  $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_I, \mathcal{S}_V)$  such that for any PPT adversary  $\mathcal{A}$ , the following experiments,  $\text{Expt}_{\mathcal{A}, \text{Real}}(1^\lambda, 1^d)$  and  $\text{Expt}_{\mathcal{A}, \text{Ideal}}(1^\lambda, 1^d)$  are indistinguishable:*

$\text{Expt}_{\mathcal{A}, \text{Real}}(1^\lambda, 1^d)$ :

1. On input the security parameter  $1^\lambda$  and a circuit depth  $1^d$ , the adversary  $\mathcal{A}$  outputs  $\mathbb{A} \in \mathbb{S}$ .
2. The challenger runs  $(\text{pk}, \text{sk}_1, \dots, \text{sk}_n) \leftarrow \text{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A})$  and provides  $\text{pk}$  to  $\mathcal{A}$ .
3.  $\mathcal{A}$  outputs an invalid (not necessarily maximal) party set  $S^* \subseteq \{P_1, \dots, P_n\}$  and messages  $\mu_1, \dots, \mu_k \in \{0, 1\}$ .
4. The challenger sends the keys  $\{\text{sk}_i\}_{i \in S^*}$  and  $\{\text{ct}_i \leftarrow \text{TFHE.Encrypt}(\text{pk}, \mu_i)\}_{i \in [k]}$  to  $\mathcal{A}$ .
5.  $\mathcal{A}$  issues a polynomial number of adaptive queries of the form  $(S \subseteq \{P_1, \dots, P_n\}, C)$  for circuits  $C : \{0, 1\}^k \rightarrow \{0, 1\}$  of depth at most  $d$ . For each query, the challenger computes  $\text{ct} \leftarrow \text{TFHE.Eval}(\text{pk}, C, \text{ct}_1, \dots, \text{ct}_k)$  and provides  $p_i \leftarrow \{\text{TFHE.PartDec}(\text{pk}, \text{sk}_i, \text{ct})\}_{i \in S}$  to  $\mathcal{A}$ .
6. At the end of the experiment,  $\mathcal{A}$  outputs a distinguishing bit  $b$ .

$\text{Expt}_{\mathcal{A}, \text{Ideal}}(1^\lambda, 1^d)$ :

1. On input the security parameter  $1^\lambda$  and a circuit depth  $1^d$ ,  $\mathcal{A}$  outputs  $\mathbb{A} \in \mathbb{S}$ .
2. The challenger runs  $(\text{pk}, \text{sk}_1, \dots, \text{sk}_n, \text{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^d, \mathbb{A})$  and sends  $\text{pk}$  to  $\mathcal{A}$ .
3.  $\mathcal{A}$  outputs an invalid party set  $S^* \subseteq \{P_1, \dots, P_n\}$  and messages  $\mu_1, \dots, \mu_k \in \{0, 1\}$ .

4. The challenger sends the keys  $\{\text{sk}_i\}_{i \in S^*}$  and  $\{\text{ct}_i \leftarrow \text{TFHE.Encrypt}(\text{pk}, \mu_i)\}_{i \in [k]}$  to  $\mathcal{A}$ .
5.  $\mathcal{A}$  issues a polynomial number of adaptive queries of the form  $(S \subseteq \{P_1, \dots, P_n\}, C)$  for circuits  $C : \{0, 1\}^k \rightarrow \{0, 1\}$  of depth at most  $d$ . For each query, the challenger provides  $\{p_i\}_{i \in S}$  computed as follows:
  - If  $C$  is queried for the first time, then initialize  $S_C = \emptyset$ .
  - If  $S \cup S_C \cup S^*$  is an invalid party set, then

$$(\text{st}', \{p_i\}_{i \in S}) \leftarrow \mathcal{S}_I(C, \{\text{ct}_1, \dots, \text{ct}_k\}, S, \text{st})$$

Else, if  $S \cup S_C \cup S^*$  is a valid party set, then

$$(\text{st}', \{p_i\}_{i \in S}) \leftarrow \mathcal{S}_V(C, \{\text{ct}_1, \dots, \text{ct}_k\}, C(\mu_1, \dots, \mu_k), S, \text{st}).$$

- Update  $S_C = S_C \cup S$  and  $\text{st} = \text{st}'$ .

6. At the end of the experiment,  $\mathcal{A}$  outputs a distinguishing bit  $b$ .

Our definition differs from [BGG+18] mainly in the ideal experiment where we define two simulators  $\mathcal{S}_I$  and  $\mathcal{S}_V$  instead of  $\mathcal{S}_2$  in [BGG+18]. Note that  $\mathcal{S}_I$  simulates the response for partial evaluation queries  $(S, C)$  if  $S$  forms an invalid set of parties along with  $S^*$  and the sets corresponding to previous queries for the circuit  $C$ . The key property of  $\mathcal{S}_I$  is that it does not need  $C(\mu_1, \dots, \mu_k)$  as its input.  $\mathcal{S}_V$  simulates the response for partial evaluation queries  $(S, C)$  when  $S$  forms a valid party set (along with  $S^*$  and the sets corresponding to previous queries for the circuit  $C$ ) and takes  $C(\mu_1, \dots, \mu_k)$  as one of its input.

## 5.1 Construction of Threshold FHE from $\{0, 1\}$ -LSSS

**Construction 1 (TFHE).** Let  $P = \{P_1, \dots, P_n\}$ . We use the following building blocks to construct a TFHE for  $P$ :

- A special fully homomorphic encryption scheme,  $\text{FHE} = (\text{FHE.Setup}, \text{FHE.Encrypt}, \text{FHE}, \text{Eval}, \text{FHE.Decrypt})$  with noise bound  $B = B(\lambda, d, q)$  and multiplicative constant 1 (Definition 5 in the full version).
- A ( $\delta$ -almost) KHPRF,  $F : \mathcal{K} \times \{0, 1\}^\lambda \rightarrow \mathbb{Z}_q$ .
- A  $\{0, 1\}$ -LSSS,  $\text{bSS} = (\text{bSS.Share}, \text{bSS.Combine})$ . We use  $T_i$  to denote the  $i$ -th partition of the share matrix  $M$ . We use  $\ell = \ell(\lambda, n)$  to denote a fixed polynomial bound on the size of the share:  $|T_i| \leq \ell$  for all  $i \in [n]$ . We let  $B_{sm}$  be the bound on the smudging noise to hide the LWE error  $e \in [-B, B]$  and  $E_{sm}$  the bound on the noise added to smudge the error in homomorphic evaluation of KHPRF.
- A collision resistant hash function  $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ .

$\text{TFHE.Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n})$ : On input the security parameter  $\lambda$ , depth bound  $d$ , and threshold access structure  $\mathbb{A}_{t,n}$ , the setup algorithm does the following:

1. Samples  $(\text{fpk}, \text{fsk}) \leftarrow \text{FHE.Setup}(1^\lambda, 1^d)$ .

2. *Secret shares*  $0 \in \mathcal{K}$  as  $(K_1, \dots, K_n) \leftarrow \text{bSS.Share}(0, \mathbb{A}_{t,n})$  and  $\text{fsk}$  as  $(\text{fsk}_1, \dots, \text{fsk}_n) \leftarrow \text{bSS.Share}(\text{fsk}, \mathbb{A}_{t,n})$ . We let  $\text{fsk}_i = \{\text{fhesk}_j\}_{j \in T_i}$  and  $K_i = \{k_j\}_{j \in T_i}$ .
3. *Outputs*  $\text{tfpk} = \text{fpk}$  and  $\text{tfsk}_i = (\text{fsk}_i, K_i)$  for all  $i \in [n]$ .

TFHE.  $\text{Encrypt}(\text{tfpk}, \mu)$ : On input the public key  $\text{tfpk}$  and input  $\mu \in \{0, 1\}$ , the encryption algorithm computes and returns  $\text{ct} = \text{FHE.Encrypt}(\text{tfpk}, \mu)$ .

TFHE.  $\text{Eval}(\text{tfpk}, C, \{\text{ct}_1, \dots, \text{ct}_k\})$ : On input the public key  $\text{tfpk}$ , a circuit  $C : \{0, 1\}^k \rightarrow \{0, 1\}$  of depth at most  $d$  and ciphertexts  $\text{ct}_1, \dots, \text{ct}_k$ , the evaluation algorithm computes and returns  $\text{ct}_C = \text{FHE.Eval}(\text{tfpk}, C, \text{ct}_1, \dots, \text{ct}_k)$ .

TFHE.  $\text{PartDec}(\text{tfpk}, \text{tfsk}_i, \text{ct})$ : On input the public key  $\text{tfpk}$ , a ciphertext  $\text{ct}$  and partial decryption key  $\text{tfsk}_i$ , the partial decryption algorithm does the following.

1. *Parse*  $\text{tfsk}_i = (\{\text{fhesk}_j\}_{j \in T_i}, \{k_j\}_{j \in T_i})$  and sample  $\{\xi_j\}_{j \in T_i} \leftarrow [-B_{sm}, B_{sm}]$  and  $\{\eta_j\}_{j \in T_i} \leftarrow [-E_{sm}, E_{sm}]$ .
2. *Compute and return*  $p_i = \{y_j = \text{FHE.decode}_0(\text{fhesk}_j, \text{ct}) + \xi_j + F(k_j, H(C)) + \eta_j\}_{j \in T_i}$ <sup>11</sup>.

TFHE.  $\text{FinDec}(\text{tfpk}, S, \{p_i\}_{i \in S})$ : On input a public key  $\text{tfpk}$ , a set  $S \subseteq [n]$ , and a set of partial decryption shares  $\{p_i\}_{i \in S}$ , it first checks if  $S \in \mathbb{A}_{t,n}$ . If no, then it outputs  $\perp$ . Else, it computes and returns

$$\mu = \text{FHE.decode}_1(\text{bSS.Combine}(\{p_i\}_{i \in S})),$$

which involves following steps: parse  $p_i = \{y_j\}_{j \in T_i}$  for each  $i \in S$  and compute a minimal valid shares set  $T \subseteq \bigcup_{i \in S} T_i$ . Then compute  $\text{FHE.decode}_1(\sum_{j \in T} y_j)$ .

**Correctness.** The correctness is the same as that in [BG<sup>+</sup>18] along with the observation that  $\text{bSS.Combine}(\{\{F(k_j, H(C))\}_{j \in T_i}\}_{i \in S}) = 0$  for all  $C$ . We provide detailed proof of correctness in the full version.

**Security and Compactness.** Compactness and semantic security follows directly from the security of underlying FHE and bSS and compactness of FHE. Please see full version for the details.

### Simulation Security

**Theorem 3.** *Assume that FHE is secure,  $F$  is a secure  $\delta$ -almost KHPRF and bSS is a secure  $\{0, 1\}$ -LSSS and  $H$  is collision resistant. Then the above construction of TFHE satisfies stronger simulation security (Definition 5).*

*Proof.* Let  $\mathcal{A}$  be any PPT adversary against the stronger simulation security of TFHE. Then we prove the above theorem via the following sequence of hybrid experiments with  $\mathcal{A}$ .

<sup>11</sup> The two smudging noises,  $\xi_j$  and  $\eta_j$  can in fact be merged together by appropriately setting the parameters.

**Hybrid<sub>0</sub>** : This is the real experiment  $\text{Expt}_{\mathcal{A}, \text{real}}(1^\lambda, 1^d)$ . On input the access structure,  $\mathbb{A}_{t,n}$ , the challenger runs  $(\text{tfpk}, \text{tfsk}_1, \dots, \text{tfsk}_n) \leftarrow \text{TFHE}.\text{Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n})$  and sends  $\text{tfpk}$  to  $\mathcal{A}$ . Then  $\mathcal{A}$  outputs a set  $S^* \subseteq [n]$  such that  $|S^*| < t$  and a set of messages  $\mu_1, \dots, \mu_k \in \{0, 1\}$ . The challenger then computes  $\text{ct}_i = \text{TFHE}.\text{Encrypt}(\text{tfpk}, \mu_i)$  for  $i \in [k]$  and returns  $\{\text{tfsk}_i\}_{i \in S^*}$  and  $\{\text{ct}_i\}_{i \in [k]}$  to  $\mathcal{A}$ . For each evaluation query  $(S, C)$  from  $\mathcal{A}$ , the challenger computes  $\text{ct}_C = \text{TFHE}.\text{Eval}(\text{tfpk}, C, \text{ct}_1, \dots, \text{ct}_k)$  and returns the following to  $\mathcal{A}$ :

$$\{p_{C,i} = \text{TFHE}.\text{PartDec}(\text{tfpk}, \text{tfsk}_i, \text{ct}_C)\}_{i \in S}.$$

Each  $p_{C,i} = \{y_{C,j}\}_{j \in T_i}$ , where  $y_{C,j} = \text{FHE}.\text{decode}_0(\text{fhesk}_j, \text{ct}_C) + F(k_j, H(C)) + \xi_{C,j} + \eta_{C,j}$ . In the following, for any  $X \subseteq [n]$ , we let  $T_X = \bigcup_{i \in X} T_i$ .

**Hybrid<sub>1</sub>**: This is the same as the previous hybrid except that for each query  $(S, C)$ , the PRF component in  $\text{TFHE}.\text{PartDec}$  is replaced with random values. In more detail, for  $i \in S \cap S^*$ ,  $p_{C,i}$  is computed as in the real world. For  $i \in S \setminus S^*$ ,  $p_{C,i}$  is computed as follows:

- If  $C$  is queried for the first time (and  $i$  is the first party in  $S \setminus S^*$ ), generate  $\{r_{C,j}\}_{j \in T_{[n] \setminus S^*}} \leftarrow \text{bSS}.\text{ShareInt}(T_{S^*}, \{F(k_j, C)\}_{j \in T_{S^*}}, 0)$ , and save them in a list  $\mathcal{L}_C$  for future iterations and queries. Else, lookup for previously saved values of  $\{r_{C,j}\}_{j \in T_i}$  in  $\mathcal{L}_C$ .
- Compute  $y_{C,j} = \text{FHE}.\text{decode}_0(\text{fhesk}_j, \text{ct}_C) + r_{C,j} + \xi_{C,j} + \eta_{C,j}$  for all  $j \in T_i$ .

**Hybrid<sub>2</sub>**: This is the same as the previous hybrid, except that for each query  $(S, C)$ , for all  $i \in S \setminus S^*$ , the PRF components in  $p_{C,i}$  are generated from bSS simulators as follows.

- If  $C$  is queried for the first time then initialize  $S_C = \emptyset$ ,  $\text{st}_C = \emptyset$ ,  $\mathcal{L}_C = \emptyset$ .
- If  $S^* \cup S_C \cup S$  is an invalid party set then generate  $(\{r_{C,j}\}_{j \in T_{S \setminus S^*}}, \text{st}'_C) \leftarrow \text{bSS}.\text{SSSiml}(T_{S^* \cup S_C}, \{r_{C,j}\}_{j \in T_{S^* \cup S_C}}, T_{S \setminus S^*}, \text{st}_C)$ . Else, if  $S^* \cup S_C \cup S$  is a valid party set then generate  $(\{r_{C,j}\}_{j \in T_{S \setminus S^*}}, \text{st}'_C) \leftarrow \text{bSS}.\text{SSSimV}(T_{S^* \cup S_C}, \{r_{C,j}\}_{j \in T_{S^* \cup S_C}}, 0, T_{S \setminus S^*}, \text{st}_C)$ .

Here, for  $j \in T_{S^*}$ ,  $r_{C,j} = F(k_j, H(C))$  and for  $j \in T_{S_C \setminus S^*}$ ,  $r_{C,j}$  is computed during previous queries for  $C$  and is stored in  $\mathcal{L}_C$ .

- Update  $S_C = S_C \cup S$  and  $\text{st}_C = \text{st}'_C$  and add  $\{r_{C,j}\}_{j \in T_{S \setminus S^*}}$  to  $\mathcal{L}_C$ .

Then compute and return  $y_{C,j} = \text{FHE}.\text{decode}_0(\text{fhesk}_j, \text{ct}) + r_{C,j} + \xi_{C,j} + \eta_{C,j}$  for all  $j \in T_{S \setminus S^*}$ .

**Hybrid<sub>3</sub>**: This is the same as the previous hybrid, except that for  $i \in S \setminus S^*$ ,  $p_{C,i} = \{y_{C,j}\}_{j \in T_i}$  is computed differently as  $y_{C,j} = \tilde{r}_{C,j} + \xi_{C,j} + \eta_{C,j}$ , where  $\tilde{r}_{C,j}$  are generated as follows:

- If  $C$  is queried for the first time then initialize  $S_C = \emptyset$ ,  $\text{st}_C = \emptyset$ ,  $\mathcal{L}_C = \emptyset$ .
- If  $S^* \cup S_C \cup S$  is an invalid party set then generate  $(\{\tilde{r}_{C,j}\}_{j \in T_{S \setminus S^*}}, \text{st}'_C) \leftarrow \text{bSS}.\text{SSSiml}(T_{S^* \cup S_C}, \{\tilde{r}_{C,j}\}_{j \in T_{S^* \cup S_C}}, T_{S \setminus S^*}, \text{st}_C)$ . Else, if  $S^* \cup S_C \cup S$  is a valid party set then generate

$$\begin{aligned} (\{\tilde{r}_{C,j}\}_{j \in T_{S \setminus S^*}}, \text{st}'_C) &\leftarrow \text{bSS}.\text{SSSimV}(T_{S^* \cup S_C}, \{\tilde{r}_{C,j}\}_{j \in T_{S^* \cup S_C}}, \\ &\lfloor q/2 \rfloor C(\mu_1, \dots, C_k), T_{S \setminus S^*}, \text{st}_C) \end{aligned}$$

Here, for  $j \in T_{S^*}$ ,  $\tilde{r}_{C,j} = \text{FHE.decode}_0(\text{fhesk}_j, \text{ct}) + F(k_j, H(C))$  and for  $j \in T_{S_C \setminus S^*}$ ,  $\tilde{r}_{C,j}$  is computed during previous queries for  $C$  and is saved in  $\mathcal{L}_C$ .

- Update  $S_C = S_C \cup S$  and  $\text{st}_C = \text{st}'_C$  and save  $\{\tilde{r}_{C,j}\}_{j \in T_{S \setminus S^*}}$  in  $\mathcal{L}_C$ .

**Hybrid<sub>4</sub>**: In this hybrid, the challenger secret shares  $0^{|\text{fsk}|}$  in place of  $\text{fsk}$  in the setup phase.

**Hybrid<sub>5</sub>**: In this hybrid,  $\text{ct}_i$  encrypts 0 instead of  $\mu_i$ .

We observe that the challenger does not use  $\text{fsk}$  or  $\mu_1, \dots, \mu_k$  to generate the secret key shares  $\{\text{fsk}_i\}_{i \in [n]}$  or to reply **PartDec** queries. Thus the challenger in **Hybrid<sub>4</sub>** corresponds to the simulator in the ideal experiment, as desired.

Due to space constraints, we argue indistinguishability of hybrids in the full version.

**Parameters.** For security and correctness, we require

- $B + \ell B_{sm} + \delta \ell + \ell E_{sm} \leq q/4$  (for correctness)
- $B/B_{sm} \leq \text{neg}(\lambda)$  and  $\delta \ell / E_{sm} \leq \text{neg}(\lambda)$  (for security).

We observe that since  $\ell$  is  $\text{poly}(\lambda)$  and  $\delta$  is a constant, our parameters are similar to those in [BGG+18]. As noted there, FHE satisfying these parameters is known from subexponential LWE assumption [GSW13, BV11], which is as hard as approximating the shortest vector with subexponential approximation factors.

*Remark 2.* In [ASY22], Agrawal *et. al* use Rényi divergence [BLL+15] based analysis to reduce the size of noise flooding in BGGJKRS threshold signature from exponential to polynomial, effectively reducing the size of modulus  $q$  to  $\text{poly}(\lambda)$ . Rényi divergence is more suitable for search based primitives, like signatures. We remark that using similar analysis as in [ASY22], size of  $B_{sm}$  and  $E_{sm}$  can also be reduced to polynomial in case of threshold signatures. However, this does not apply for threshold FHE and UT which are indistinguishability based primitives.

## 6 Universal Thresholdizer

We recall the definition of universal thresholdizer from [BGG+18] in the full version. Here we define our stronger security notion for universal thresholdizer needed to prove stronger security for the primitives thresholdized using it - for example, threshold signatures and threshold CCA-PKE.

**Definition 6 (UT Stronger Security).** *We say that a UT scheme satisfies (stronger) security if for all  $\lambda$ , and depth bound  $d$ , the following holds. There exists a stateful PPT algorithm  $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_I, \mathcal{S}_V)$  such that for all PPT adversary  $\mathcal{A}$ , we have that the following experiments  $\text{Exp}_{\mathcal{A}, \text{UT}, \text{Real}}(1^\lambda, 1^d)$  and  $\text{Exp}_{\mathcal{A}, \text{UT}, \text{Ideal}}(1^\lambda, 1^d)$  are computationally indistinguishable:*

$\text{Exp}_{\mathcal{A}, \text{UT}, \text{Real}}(1^\lambda, 1^d)$ :

1. On input the security parameter  $1^\lambda$ , and circuit depth  $1^d$ , the adversary  $\mathcal{A}$  outputs an access structure  $\mathbb{A}_{t,n}$ , and a message  $x \in \{0, 1\}^k$ .
2. The challenger runs  $(\text{pp}, s_1, \dots, s_n) \leftarrow \text{UT.Setup}(1^\lambda, 1^d, \mathbb{A}_{t,n}, x)$  and provides  $\text{pp}$  to  $\mathcal{A}$ .
3.  $\mathcal{A}$  outputs an invalid party set  $S^* \subset \{P_1, \dots, P_n\}$  for  $\mathbb{A}_{t,n}$ .
4. The challenger provides the shares  $\{s_i\}_{i \in S^*}$  to  $\mathcal{A}$ .
5.  $\mathcal{A}$  issues a polynomial number of adaptive queries of the form  $(S \subseteq \{P_1, \dots, P_n\}, C)$  for circuits  $C : \{0, 1\}^k \rightarrow \{0, 1\}$  of depth at most  $d$ . For each query, the challenger provides  $\{y_i \leftarrow \text{UT.Eval}(\text{pp}, s_i, C)\}_{i \in S}$  to  $\mathcal{A}$ .
6. At the end of the experiment,  $\mathcal{A}$  outputs a distinguishing bit  $b$ .

$\text{Expt}_{\mathcal{A}, \text{UT}, \text{Ideal}}(1^\lambda, 1^d)$ :

1. On input the security parameter  $1^\lambda$ , and circuit depth  $1^d$ , the adversary  $\mathcal{A}$  outputs an access structure  $\mathbb{A}_{t,n}$ , and a message  $x \in \{0, 1\}^k$ .
2. The challenger runs  $(\text{pp}, s_1, \dots, s_n, \text{st}) \leftarrow \mathcal{S}_1(1^\lambda, 1^d, \mathbb{A}_{t,n})$  and sends  $\text{pp}$  to  $\mathcal{A}$ .
3.  $\mathcal{A}$  outputs an invalid party set  $S^* \subset \{P_1, \dots, P_n\}$  for  $\mathbb{A}_{t,n}$ .
4. The challenger provides the shares  $\{s_i\}_{i \in S^*}$  to  $\mathcal{A}$ .
5.  $\mathcal{A}$  issues polynomial number of adaptive queries of the form  $(S \subseteq \{P_1, \dots, P_n\}, C)$  for circuits  $C : \{0, 1\}^k \rightarrow \{0, 1\}$  of depth at most  $d$ . For each query, the challenger provides  $\{y_i\}_{i \in S}$  computed as follows:
  - If  $C$  is queried for the first time, then initialize  $S_C = S$ , else  $S_C = S_C \cup S$ .
  - If  $S_C \cup S^*$  is an invalid party set, then  $(\{y_i\}_{i \in S}, \text{st}') \leftarrow \mathcal{S}_I(\text{pp}, C, S, \text{st})$ .
  - Else, if  $S_C \cup S^*$  is a valid party set, then  $(\{y_i\}_{i \in S}, \text{st}') \leftarrow \mathcal{S}_V(\text{pp}, C, C(x), S, \text{st})$ .
  - Update  $\text{st} = \text{st}'$ .
6. At the end of the experiment,  $\mathcal{A}$  outputs a distinguishing bit  $b$ .

*Remark 3.* The above definition differs from the security definition in [BGG+18] in the ideal experiment. In the weaker notion of [BGG+18], in response to any query  $(S, C)$ , the UT simulator takes  $C(x)$  as input, irrespective of whether or not the set  $S$  forms a valid party set along with  $S^*$  and sets corresponding to the previous queries for the circuit  $C$ . In our definition, we make this distinction - we define two simulators  $\mathcal{S}_I$  and  $\mathcal{S}_V$ .  $\mathcal{S}_I$  is used in case of invalid set and does not take  $C(x)$  as input.  $\mathcal{S}_V$  is used when  $S$  forms a valid set (along with  $S^*$  and sets in previous queries for  $C$ ) and takes  $C(x)$  as input.

[BGG+18] uses a TFHE and NIZK with preprocessing (PZK) to construct a universal thresholdizer. Below we show that if the TFHE scheme satisfies the semantic security and *stronger* simulation security (Definition 5), then the construction of UT in [BGG+18] satisfies the stronger security (Definition 6). Due to space constraints, we recall the construction from [BGG+18] in the full version.

**Theorem 4.** *Suppose TFHE satisfies semantic security ([BGG+18, Definition 5.5]) and stronger simulation security (Definition 5), PZK is a zero knowledge proof system with pre-processing that satisfies zero-knowledge, and  $C$  is a non-interactive commitment scheme that satisfies computational hiding. Then, the universal thresholdizer scheme by Boneh et al. [BGG+18, Construction 7.7] satisfies the stronger security (Definition 6).*



*Proof.* The proof for the above theorem is similar to the proof of [BGG+18, Theorem 7.11]. Here we provide a sketch of the proof and refer to the full version for complete proof. The main difference is in the description of  $\text{Hybrid}_3$ , where for each query  $(S, C)$ , the challenger does the following:

1. If  $C$  is queried for the first time, initialize  $S_C = \emptyset$ .
2. If  $S \cup S_C \cup S^*$  is an invalid party set then the partial evaluations  $\{p_i^*\}_{i \in S}$  are generated using  $\mathcal{S}_I$  simulator for TFHE.  
Else, if  $S \cup S_C \cup S^*$  is a valid party set then the partial evaluations  $\{p_i^*\}_{i \in S}$  are generated using  $\mathcal{S}_V$  simulator for TFHE.
3. Update  $S_C = S_C \cup S$ .

Here  $S^*$  is the set of corrupted parties and as per our definition of UT security,  $S^*$  is an invalid party set, but not necessarily *maximally* invalid.

The indistinguishability of  $\text{Hybrid}_3$  from  $\text{Hybrid}_2$  follows directly from the stronger simulation security of TFHE (Definition 5).

## 7 Applications

In this section we revisit the application of universal thresholdizer in thresholdizing different crypto primitives as considered in [BGG+18]. In particular we define and construct threshold signatures, threshold CCA PKE and function secret sharing with stronger security properties. We note that the constructions for these primitives using universal thresholdizer is the same as in [BGG+18]. Our contribution lies in (defining and) proving stronger security for these primitives assuming that the underlying universal thresholdizer satisfies the stronger security as defined in Sect. 6.

### 7.1 Threshold Signatures

We recall the definition of threshold signatures and its desired properties in the full version. Below we describe selective unforgeability from [BGG+18] and its stronger notion from [BTZ22].

**Definition 7 (Selective Unforgeability [BGG+18]).** A TS scheme is unforgeable if for all PPT adversary  $\mathcal{A}$ , the probability of winning in the following experiment,  $\text{Expt}_{\mathcal{A}, \text{TS}, \text{uf}}(1^\lambda)$  is  $\text{neg}(\lambda)$ .

1. On input the security parameter  $\lambda$  and an access structure  $\mathbb{A}_{t,n}$ , the challenger runs the  $\text{TS.KeyGen}(1^\lambda, \mathbb{A}_{t,n})$  algorithm and generates public parameters  $\text{pp}$ , verification key  $\text{vk}$  and set of  $n$  key shares  $\{\text{sk}_i\}_{i=1}^n$ . It sends  $\text{pp}$  and  $\text{vk}$  to  $\mathcal{A}$ .
2.  $\mathcal{A}$  then outputs a maximally invalid party set  $S^* \subset [n]$ , i.e.  $|S^*| = t - 1$ , requesting key shares  $\text{sk}_i$  for  $i \in S^*$ .
3. Challenger provides the set of keys  $\{\text{sk}_i\}_{i \in S^*}$  to  $\mathcal{A}$ .
4. Adversary  $\mathcal{A}$  issues polynomial number of adaptive queries of the form  $(m, i)$ , where  $i \in [n] \setminus S^*$ , to get partial signature  $\sigma_i$  on  $m$ . For each query the challenger computes  $\sigma_i$  as  $\text{TS.PartSign}(\text{pp}, \text{sk}_i, m)$  and provides it to  $\mathcal{A}$ .

5. At the end of the experiment,  $\mathcal{A}$  outputs a message-signature pair  $(m^*, \sigma^*)$ . The adversary wins if the following conditions hold:
- (a)  $m^*$  was never queried as a signing query.
  - (b)  $\text{TS.Verify}(\text{vk}, m^*, \sigma^*) = \text{accept}$ .

**Definition 8 (Stronger Selective Unforgeability [BTZ22]).** In the stronger definition, the set  $S^*$  of corrupted parties is not necessarily maximally invalid, i.e.  $|S^*| < t$ . Another and the main difference is in the conditions under which the adversary wins as defined below:

The adversary is allowed to issue partial signatures on the challenge message  $m^*$ , and wins if the following conditions hold:

1. Let  $S_{m^*} = \{i : (m^*, i) \text{ was queried as a signing query}\}$ . Then  $|S^* \cup S_{m^*}| < t$ .
2.  $\text{TS.Verify}(\text{vk}, m^*, \sigma^*) = \text{accept}$ .

Similar to [BGG+18], we construct a threshold signature scheme from a universal thresholdizer and a signature scheme.

**Construction 2 (Construction 8.16 in [BGG+18]).** The construction  $\text{TS} = (\text{TS.KeyGen}, \text{TS.PartSign}, \text{TS.PartSignVerify}, \text{TS.Combine}, \text{TS.Verify})$  uses a signature scheme  $\text{SignScheme} = (\text{SGen}, \text{Sign}, \text{Verify})$  and a universal thresholdizer  $\text{UT} = (\text{UT.Setup}, \text{UT.Eval}, \text{UT.Verify}, \text{UT.Combine})$ .

- $\text{TS.KeyGen}(1^\lambda, \mathbb{A}_{t,n}) \rightarrow (\text{pp}, \text{vk}, \{\text{sk}_i\}_{i=1}^n)$ . First it invokes  $\text{SGen}(1^\lambda)$  to obtain a pair  $(\text{pk}, \text{sk})$  and it sets  $\text{vk} := \text{pk}$ . Then it runs  $(\text{upp}, \{\text{usk}_i\}_{i=1}^n) \leftarrow \text{UT.Setup}(1^\lambda, \mathbb{A}_{t,n}, \text{sk})$ . Sets  $\text{pp} = \text{upp}$  and  $\{\text{sk}_i = \text{usk}_i\}_{i=1}^n$ .
- $\text{TS.PartSign}(\text{pp}, \text{sk}_i, m) \rightarrow \sigma_i$ . On input the public parameters  $\text{pp}$ , a partial signing key  $\text{sk}_i$  and a message  $m$ , the partial signing algorithm outputs  $\text{UT.Eval}(\text{pp}, \text{sk}_i, C_m)$  where the circuit  $C_m$  is defined as

$$C_m(\text{sk}) := \text{Sign}(\text{sk}, m).$$

- $\text{TS.PartSignVerify}(\text{pp}, m, \sigma_i) \rightarrow \text{accept/reject}$ . On input the public parameters  $\text{pp}$ , message  $m$ , and a partial signature  $\sigma_i$ , the partial signature verification algorithm outputs  $\text{UT.Verify}(\text{pp}, \sigma_i, C_m)$ .
- $\text{TS.Combine}(\text{pp}, \{\sigma_i\}_{i \in S}) \rightarrow \sigma_m$ . On input the public parameters  $\text{pp}$ , and a set of partial signatures  $\{\sigma_i\}_{i \in S}$ , the signature combining algorithm outputs  $\text{UT.Combine}(\text{pp}, \{\sigma_i\}_{i \in S})$ .
- $\text{TS.Verify}(\text{vk}, m, \sigma_m) \rightarrow \text{accept/reject}$ . On input the signature verification key  $\text{vk} = \text{pk}$ , a message  $m$ , and a signature  $\sigma$ , the verification algorithm outputs  $\text{Verify}(\text{pk}, m, \sigma)$ .

**Theorem 5.** If the universal thresholdizer  $\text{UT}$  satisfies the stronger security notion (Definition 6) and  $\text{SignScheme}$  is a signature scheme that satisfies unforgeability, then the construction above (Construction 2) satisfies the Stronger Selective Unforgeability (Definition 8).

*Proof.* We prove the above theorem via the following hybrids. We start with  $\text{Hybrid}_0$  which is the experiment  $\text{Expt}_{\mathcal{A}, \text{TS}, uf}(1^\lambda)$  for the Construction 2.

$\text{Hybrid}_1$ . Note that since UT is secure with respect to Definition 6, there exists a stateful PPT algorithm  $\text{UT.S} = (\text{UT.S}_1, \text{UT.S}_I, \text{UT.S}_V)$  which can simulate answer to  $\text{UT.Eval}(\text{pp}, \text{sk}_i, \cdot)$  queries. We define  $\text{Hybrid}_1$  that is similar to  $\text{Hybrid}_0$  except that for the challenge queries of the form  $(m, i)$  made by  $\mathcal{A}$ , the challenger uses  $\text{UT.S} = (\text{UT.S}_1, \text{UT.S}_I, \text{UT.S}_V)$  to generate partial signatures. It is straightforward to show these two hybrids are indistinguishable because UT is secure with respect to Definition 6.

Furthermore, we show that the advantage of  $\mathcal{A}$  in  $\text{Hybrid}_1$  is negligible. Let us assume that the advantage of  $\mathcal{A}$  in  $\text{Hybrid}_1$  is  $\epsilon$ . We define a reduction adversary  $\mathcal{B}$  to attack the unforgeability of the underlying signature scheme  $\text{SignScheme} = (\text{SGen}, \text{Sign}, \text{Verify})$ . Namely, the adversary  $\mathcal{B}$  after receiving the public key  $\text{pk}$  from its challenger, sets  $\text{vk} = \text{pk}$ . It runs  $(\text{pp}, \text{sk}_1, \dots, \text{sk}_n, \text{st}) \leftarrow \text{UT.S}_1(1^\lambda, 1^d, \mathbb{A}_{t,n})$  and provides  $\text{pp}$  to  $\mathcal{A}$ .

- When  $\mathcal{A}$  outputs an invalid party set  $S^* \subset [n]$ , the adversary  $\mathcal{B}$  provides the set of keys  $\{\text{sk}_i\}_{i \in S^*}$  to  $\mathcal{A}$ .
- Adversary  $\mathcal{A}$  issues polynomial number of adaptive queries of the form  $(m, i)$ , where  $i \in [n] \setminus S^*$  to get partial signature  $\sigma_i$  for  $m$ . For each query the adversary  $\mathcal{B}$  computes  $\sigma_i$  computed as follows:
  - If  $m$  is queried for the first time, then initialize  $S_m = S$ , else  $S_m = S_m \cup S$ .
  - If  $S_m \cup S^*$  is an invalid party set, then  $(\{\sigma_i\}_{i \in S}, \text{st}') \leftarrow \text{UT.S}_I(\text{pp}, C_m, S, \text{st})$
  - Else, if  $S_m \cup S^*$  is a valid party set, then it queries  $m$  to its challenger to receive a signature  $\sigma_m$  on  $m$ , and returns  $(\{\sigma_i\}_{i \in S}, \text{st}') \leftarrow \text{UT.S}_V(\text{pp}, C_m, \sigma_m, S, \text{st})$ .
  - Update  $\text{st} = \text{st}'$ .
- When at the end of the experiment,  $\mathcal{A}$  outputs a message-signature pair  $(m^*, \sigma^*)$ , the adversary  $\mathcal{B}$  returns  $(m^*, \sigma^*)$  as a forgery for  $\text{SignScheme}$ .

We observe that  $(m^*, \sigma^*)$  is a valid forgery by  $\mathcal{B}$ . This is because  $|S^* \cup S_{m^*}| < t$  due to validity of  $\mathcal{A}$ 's forgery. Hence,  $\mathcal{B}$  would never have asked a signature on  $m^*$  to its challenger to answer any partial signature query on  $m^*$  by  $\mathcal{A}$ . It is clear that if  $\mathcal{A}$  wins with  $\epsilon$  advantage, then the advantage of  $\mathcal{B}$  in breaking the unforgeability of  $\text{SignScheme}$  is also  $\epsilon$ . This finishes the proof.

## 7.2 Threshold CCA PKE

We recall the definition of CCA threshold PKE and its correctness in the full version. Below we recall its security from [BGG+18] and define a stronger security, similar to stronger security of threshold signature.

**Definition 9 (Security [BGG+18]).** A TPKE scheme for  $\mathbb{A}_{t,n}$  is said to satisfy CCA security if for all  $\lambda$ , the following holds: for all PPT adversary  $\mathcal{A}$ , following experiments,  $\text{Expt}_{\mathcal{A}, \text{TPKE}}^0(1^\lambda)$  and  $\text{Expt}_{\mathcal{A}, \text{TPKE}}^1(1^\lambda)$  are computationally indistinguishable.

$$\text{Expt}_{\mathcal{A}, \text{TPKE}}^b(1^\lambda)$$

1. On input a security parameter  $\lambda$ , and a threshold access structure  $\mathbb{A}_{t,n}$ , the challenger runs  $(\text{pp}, \text{ek}, \text{sk}_1, \dots, \text{sk}_n) \leftarrow \text{TPKE.KeyGen}(1^\lambda, \mathbb{A}_{t,n})$  and provides  $\text{pp}$  and  $\text{ek}$  to  $\mathcal{A}$ .
2.  $\mathcal{A}$  outputs a maximal invalid party set,  $S^* \subset [n]$ , that is,  $|S^*| = t - 1$ .
3. The challenger provides the set of secret keys,  $\{\text{sk}_i\}_{i \in S^*}$  to  $\mathcal{A}$ .
4.  $\mathcal{A}$  issues a polynomial number of adaptive decryption queries of the form  $(\text{ct}, i)$ , where  $i \notin S^*$ .
5. The challenger computes  $s_i \leftarrow \text{TFHE.PartDec}(\text{pp}, \text{sk}_i, \text{ct})$  and sends  $s_i$  to  $\mathcal{A}$ .
6.  $\mathcal{A}$  outputs a pair of challenge messages  $(m_0^*, m_1^*)$ .
7. The challenger computes  $\text{ct}^* \leftarrow \text{TFHE.Encrypt}(\text{ek}, m_b^*)$  and sends  $\text{ct}^*$  to  $\mathcal{A}$ .
8.  $\mathcal{A}$  continues issuing a polynomial number of adaptive decryption queries. However, the adversary is not allowed to issue a decryption query on the challenge ciphertext  $\text{ct}^*$ .
9. At the end of the experiment,  $\mathcal{A}$  outputs a guess bit  $b'$ .

Similar to the stronger security of threshold signature, we define the stronger security for TPKE, where the adversary is allowed to issue partial decryption query on the challenge ciphertext as well, as long as it is for an invalid set of parties over all such queries.

**Definition 10 (Stronger Security TPKE).**

In the stronger definition, the set  $S^*$  of corrupted parties is not necessarily maximally invalid, i.e.  $|S^*| < t$ . Another, and the main difference is in the admissibility condition for  $\mathcal{A}$  as follows -  $\mathcal{A}$  can issue queries of the form  $(\text{ct}^*, i)$ . Let  $S_{\text{ct}^*} = \{i : \mathcal{A} \text{ issued decryption query as } (\text{ct}^*, i)\}$ . Then,  $S^* \cup S_{\text{ct}^*}$  must be an invalid set, i.e.  $|S^* \cup S_{\text{ct}^*}| < t$ .

Similar to [BGG+18], we construct a threshold public-key encryption scheme from a universal thresholdizer and a public-key encryption.

**Construction 3 (Construction 8.29 in [BGG+18]).** The construction  $\text{TPKE} = (\text{TPKE.KeyGen}, \text{TPKE.Encrypt}, \text{TPKE.PartDec}, \text{TPKE.Combine})$  uses a public key encryption scheme  $\text{PKE} = (\text{PKE.Gen}, \text{Enc}, \text{Dec})$  and a universal thresholdizer  $\text{UT} = (\text{UT.Setup}, \text{UT.Eval}, \text{UT.Verify}, \text{UT.Combine})$ .

- $\text{TPKE.KeyGen}(1^\lambda, \mathbb{A}_{t,n}) \rightarrow (\text{pp}, \text{ek}, \text{sk}_1, \dots, \text{sk}_n)$ . First it invokes  $\text{PKE.Gen}(1^\lambda)$  to obtain a pair  $(\text{pk}, \text{sk})$  and it sets  $\text{ek} := \text{pk}$ . Then it runs  $(\text{upp}, \{\text{usk}_i\}_{i \in [n]}) \leftarrow \text{UT.Setup}(1^\lambda, \mathbb{A}_{t,n}, \text{sk})$ , and sets  $\text{pp} = \text{upp}$  and  $\{\text{sk}_i = \text{usk}_i\}_{i=1}^n$ .
- $\text{TPKE.Encrypt}(\text{ek}, m) \rightarrow \text{ct}$ . The encryption algorithm takes as input a message  $m \in \mathcal{M}$  and the encryption key  $\text{ek}$  and outputs a ciphertext  $\text{ct} \leftarrow \text{Enc}(\text{pk}, m)$ .
- $\text{TPKE.PartDec}(\text{pp}, \text{sk}_i, \text{ct}) \rightarrow m_i$ . The partial decryption algorithm takes as input the public parameters,  $\text{pp}$ , decryption key share  $\text{sk}_i$  and a ciphertext  $\text{ct}$  and outputs  $m_i \leftarrow \text{UT.Eval}(\text{pp}, \text{sk}_i, C_{\text{ct}})$  where the circuit  $C_{\text{ct}}$  is defined as

$$C_{\text{ct}}(\text{sk}) := \text{Dec}(\text{ct}, \text{sk}).$$

- TPKE.Combine(pp,  $\{m_i\}_{i \in S}$ )  $\rightarrow m'$ . The combining algorithm takes the public parameters, pp and set of partially decrypted messages  $\{m_i\}_{i \in S}$  and outputs  $m' \leftarrow$  UT.Combine(pp,  $\{m_i\}_{i \in S}$ ).

**Theorem 6.** *If the universal thresholdizer UT satisfies the stronger security definition (Definition 6) and PKE is CCA secure, then the construction above (Construction 3) satisfies the stronger security Definition 10.*

*Proof.* We prove the above theorem via the following hybrids. For simplicity, we consider a modified but equivalent version of  $\text{Expt}_{\mathcal{A}, \text{TPKE}}^b(1^\lambda)$  in which the advantage of the adversary is the probability that  $b' = b$  when  $b$  is chosen uniformly random by the challenger. We start with Hybrid<sub>0</sub> which is the (modified) experiment  $\text{Expt}_{\mathcal{A}, \text{TPKE}}^b(1^\lambda)$  for the Construction 2.

Hybrid<sub>1</sub>. Note that since UT is secure with respect to Definition 6, there exists a stateful PPT algorithm  $\text{UT.S} = (\text{UT.S}_1, \text{UT.S}_I, \text{UT.S}_V)$  which can simulate answer to  $\text{UT.Eval}(\text{pp}, \text{sk}_i, \cdot)$  queries. We define Hybrid<sub>1</sub> that is similar to Hybrid<sub>0</sub> except for the challenge decryption queries of the form  $(\text{ct}, i)$  made by  $\mathcal{A}$ . For the challenge queries, the challenger uses  $\text{UT.S} = (\text{UT.S}_1, \text{UT.S}_I, \text{UT.S}_V)$  to answer. It is straightforward to show these two hybrids are indistinguishable because UT is secure with respect to Definition 6.

Furthermore, we show that the advantage of  $\mathcal{A}$  in Hybrid<sub>1</sub> is at most  $1/2 + \text{neg}$ . Let us assume that the advantage of  $\mathcal{A}$  in Hybrid<sub>1</sub> is  $\epsilon$ . We define a reduction adversary  $\mathcal{B}$  to attack the CCA security of the underlying public-key encryption scheme  $\text{PKE} = (\text{PKE.Gen}, \text{Enc}, \text{Dec})$ . Namely, the adversary  $\mathcal{B}$  after receiving the public key  $\text{pk}$  from its challenger, it sets  $\text{ek} = \text{pk}$ , it runs  $(\text{pp}, \text{sk}_1, \dots, \text{sk}_n, \text{st}) \leftarrow \text{UT.S}_1(1^\lambda, 1^d, \mathbb{A}_{t,n})$  and provides  $\text{pp.ek}$  to  $\mathcal{A}$ .

- When  $\mathcal{A}$  outputs an invalid party set  $S^* \subseteq [n]$ , the adversary  $\mathcal{B}$  provides the set of keys  $\{\text{sk}_i\}_{i \in S^*}$  to  $\mathcal{A}$ .
- Adversary  $\mathcal{A}$  issues a polynomial number of adaptive decryption queries of the form  $(\text{ct}, i)$ , where  $i \notin S^*$ . For each query the adversary  $\mathcal{B}$  computes  $m_i$  computed as follows:
  - If  $\text{ct}$  is queried for the first time, then initialize  $S_{\text{ct}} = S$ , else  $S_{\text{ct}} = S_{\text{ct}} \cup S$ .
  - If  $S_{\text{ct}} \cup S^*$  is an invalid party set, then  $(\{m_i\}_{i \in S}, \text{st}') \leftarrow \text{UT.S}_I(\text{pp}, C_{\text{ct}}, S, \text{st})_{i \in S}$
  - Else, if  $S_{\text{ct}} \cup S^*$  is a valid party set, then it queries  $\text{ct}$  to its challenger to receive a decryption  $m_{\text{ct}}$ , and  $(\{m_i\}_{i \in S}, \text{st}') \leftarrow \text{UT.S}_V(\text{pp}, C_{\text{ct}}, m_{\text{ct}}, S, \text{st})_{i \in S}$ .
  - Update  $\text{st} = \text{st}'$ .
- When  $\mathcal{A}$  outputs a pair of challenge messages  $(m_0^*, m_1^*)$ , the adversary  $\mathcal{B}$  forwards it to its challenger and receives  $\text{ct}^* \leftarrow \text{TFHE.Encrypt}(\text{ek}, m_b^*)$ , where  $b$  is the challenge bit of PKE challenger.  $\mathcal{B}$  sends  $\text{ct}^*$  to  $\mathcal{A}$ .
- $\mathcal{A}$  continues issuing a polynomial number of adaptive decryption queries  $(\text{ct}, i)$ . The adversary  $\mathcal{A}$  is allowed to issue a decryption query on the challenge ciphertext  $\text{ct}^*$  up to the gap between the threshold value and the number of

- corrupted parties. The adversary  $\mathcal{B}$  answers similar to pre-challenge queries above, unless, when  $\mathcal{A}$  queries  $\text{ct}^*$ . For  $\text{ct}^*$ , the adversary  $\mathcal{B}$  stops and returns  $\perp$  whenever  $S_{\text{ct}^*} \cup S^*$  is a valid party set. Note that as long as  $S_{\text{ct}^*} \cup S^*$  is an invalid party set,  $\mathcal{B}$  uses  $\text{UT.S}_I$ , which does not need the decryption of  $\text{ct}^*$ .
- When at the end of the experiment,  $\mathcal{A}$  outputs a bit  $b'$ , the adversary  $\mathcal{B}$  returns  $b'$  as its output.

It is clear that the advantage of  $\mathcal{B}$  in breaking the CCA security of PKE is  $\epsilon$ . This finishes the proof since by the CCA security of PKE,  $\epsilon \leq 1/2 + \text{negligible}$ .

**Acknowledgments.** Ehsan Ebrahimi is supported by the Luxembourg National Research Fund under the Junior CORE project QSP (C22/IS/17272217/QSP/Ebrahimi).

## References

- [ABV+12] Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *PKC*, volume 7293 of *LNCS*, pages 280–297. Springer, Berlin, Heidelberg, 2012. [https://doi.org/10.1007/978-3-642-30057-8\\_17](https://doi.org/10.1007/978-3-642-30057-8_17).
- [AMMR18] Shashank Agrawal, Payman Mohassel, Pratyay Mukherjee, and Peter Rindal. Discrete Distributed symmetric-key encryption. In *ACM-CCS*, pages 1993–2010. ACM, 2018. <https://doi.org/10.1145/3243734.3243774>.
- [ASY22] Shweta Agrawal, Damien Stehlé, and Anshu Yadav. Round-optimal lattice-based threshold signatures, revisited. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:20, 2022. <https://doi.org/10.4230/LIPIcs.ICALP.2022.8>.
- [BCK+22] Mihir Bellare, Elizabeth C. Crites, Chelsea Komlo, Mary Maller, Stefano Tessaro, and Chenzhi Zhu. Better than advertised security for non-interactive threshold signatures. In *CRYPTO*, volume 13510 of *LNCS*, pages 517–550. Springer, 2022. [https://doi.org/10.1007/978-3-031-15985-5\\_18](https://doi.org/10.1007/978-3-031-15985-5_18).
- [BGG+18] Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter MR Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In *CRYPTO*, volume 10991 of *LNCS*, pages 565–596. Springer, Cham, 2018. [https://doi.org/10.1007/978-3-319-96884-1\\_19](https://doi.org/10.1007/978-3-319-96884-1_19).
- [BLL+15] Shi Bai, Adeline Langlois, Tancrede Lepoint, Damien Stehlé, and Ron Steinfeld. Improved security proofs in lattice-based cryptography: Using the rényi divergence rather than the statistical distance. In *ASIACRYPT*, pages 3–24. Springer Berlin Heidelberg, 2015. [https://doi.org/10.1007/978-3-662-48797-6\\_1](https://doi.org/10.1007/978-3-662-48797-6_1).
- [BLMR13] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *CRYPTO*, volume 8042 of *LNCS*, pages 410–428. Springer, Berlin, Heidelberg, 2013. [https://doi.org/10.1007/978-3-642-40041-4\\_23](https://doi.org/10.1007/978-3-642-40041-4_23).

- [BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptol.*, 17(4):297–319, 2004. <https://doi.org/10.1007/s00145-004-0314-9>.
- [Bol02] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC*, pages 31–46. Springer, 2002. [https://doi.org/10.1007/3-540-36288-6\\_3](https://doi.org/10.1007/3-540-36288-6_3).
- [Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *PKC*, pages 31–46. Springer, 2003. [https://doi.org/10.1007/3-540-36288-6\\_3](https://doi.org/10.1007/3-540-36288-6_3).
- [BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In *CRYPTO*, volume 8616 of *LNCS*, pages 353–370. Springer, Berlin, Heidelberg, 2014. [https://doi.org/10.1007/978-3-662-44371-2\\_20](https://doi.org/10.1007/978-3-662-44371-2_20).
- [BTZ22] Mihir Bellare, Stefano Tessaro, and Chenzhi Zhu. Stronger security for non-interactive threshold signatures: Bls and frost. *Cryptology ePrint Archive*, 2022.
- [BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011. <https://doi.org/10.1137/120868669>.
- [CCK23] Jung Hee Cheon, Wonhee Cho, and Jiseung Kim. Improved universal thresholdizer from iterative shamir secret sharing. *Cryptology ePrint Archive*, Paper 2023/545, 2023.
- [CG99] Ran Canetti and Shafi Goldwasser. An Efficient *Threshold* Public Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack. In *EUROCRYPT*, volume 1592 of *LNCS*, pages 90–106. Springer, 1999. [https://doi.org/10.1007/3-540-48910-X\\_7](https://doi.org/10.1007/3-540-48910-X_7).
- [DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In *CRYPTO*, volume 435 of *LNCS*, pages 307–315. Springer, 1989. [https://doi.org/10.1007/0-387-34805-0\\_28](https://doi.org/10.1007/0-387-34805-0_28).
- [DOTT21] Ivan Damgård, Claudio Orlandi, Akira Takahashi, and Mehdi Tibouchi. Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In *PKC*, pages 99–130, Cham, 2021. Springer International Publishing. [https://doi.org/10.1007/978-3-030-75245-3\\_5](https://doi.org/10.1007/978-3-030-75245-3_5).
- [dPKM+24] Rafael del Pino, Shuichi Katsumata, Mary Maller, Fabrice Mouhartem, Thomas Prest, and Markku-Juhani Saarienen. Threshold raccoon: Practical threshold signatures from standard lattice assumptions. In *EUROCRYPT*, pages 219–248. Springer, 2024. [https://doi.org/10.1007/978-3-031-58723-8\\_8](https://doi.org/10.1007/978-3-031-58723-8_8).
- [GKS24] Kamil Doruk Gur, Jonathan Katz, and Tjerand Silde. Two-round threshold lattice-based signatures from threshold homomorphic encryption. In Markku-Juhani Saarienen and Daniel Smith-Tone, editors, *Post-Quantum Cryptography*, pages 266–300. Springer, 2024. [https://doi.org/10.1007/978-3-031-62746-0\\_12](https://doi.org/10.1007/978-3-031-62746-0_12).
- [GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *CRYPTO*, volume 8042 of *LNCS*, pages 75–92. Springer Berlin Heidelberg, 2013. [https://doi.org/10.1007/978-3-642-40041-4\\_5](https://doi.org/10.1007/978-3-642-40041-4_5).
- [GWW+13] Yuanju Gan, Lihua Wang, Licheng Wang, Ping Pan, and Yixian Yang. Efficient construction of cca-secure threshold pke based on hashed diffie-hellman assumption. *The Computer Journal*, 56, 2013.

- [KG20] Chelsea Komlo and Ian Goldberg. FROST: flexible round-optimized schnorr threshold signatures. In *SAC*, LNCS, pages 34–65. Springer, 2020. [https://doi.org/10.1007/978-3-030-81652-0\\_2](https://doi.org/10.1007/978-3-030-81652-0_2).
- [Kim20] Sam Kim. Key-homomorphic pseudorandom functions from lwe with small modulus. In *EUROCRYPT*, volume 12106 of *LNCS*, pages 576–607. Springer, Cham, 2020. [https://doi.org/10.1007/978-3-030-45724-2\\_20](https://doi.org/10.1007/978-3-030-45724-2_20).
- [MPS+02] Keith M. Martin, Josef Pieprzyk, Reihaneh Safavi-Naini, Huaxiong Wang, and Peter R. Wild. Threshold MACs. In *ICISC*, volume 2587 of *LNCS*, pages 237–252. Springer, 2002. [https://doi.org/10.1007/3-540-36552-4\\_17](https://doi.org/10.1007/3-540-36552-4_17).
- [MS23] Daniele Micciancio and Adam Suhl. Simulation-secure threshold PKE from LWE with polynomial modulus. *Cryptology ePrint Archive*, Paper 2023/1728, 2023.
- [NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdc. In *EUROCRYPT*, volume 1592 of *LNCS*, pages 327–346. Springer, Berlin, Heidelberg, 1999. [https://doi.org/10.1007/3-540-48910-X\\_23](https://doi.org/10.1007/3-540-48910-X_23).
- [Sho00] Victor Shoup. Practical threshold signatures. In *EUROCRYPT*, volume 1807 of *LNCS*, pages 207–220. Springer, 2000. [https://doi.org/10.1007/3-540-45539-6\\_15](https://doi.org/10.1007/3-540-45539-6_15).



# **Isogeny-Based Cryptography**



# Ideal-to-Isogeny Algorithm Using 2-Dimensional Isogenies and Its Application to SQIsign

Hiroshi Onuki<sup>1</sup>   and Kohei Nakagawa<sup>2</sup>

<sup>1</sup> The University of Tokyo, Tokyo, Japan  
onuki@mist.i.u-tokyo.ac.jp

<sup>2</sup> NTT Social Informatics Laboratories, Tokyo, Japan  
kohei.nakagawa@ntt.com

**Abstract.** The Deuring correspondence is a correspondence between supersingular elliptic curves and quaternion orders. Under this correspondence, an isogeny between elliptic curves corresponds to a quaternion ideal. This correspondence plays an important role in isogeny-based cryptography and several algorithms to compute an isogeny corresponding to a quaternion ideal (ideal-to-isogeny algorithms) have been proposed. In particular, SQIsign is a signature scheme based on the Deuring correspondence and uses an ideal-to-isogeny algorithm. In this paper, we propose a novel ideal-to-isogeny algorithm using isogenies of dimension 2. Our algorithm is based on Kani's reducibility theorem, which gives a connection between isogenies of dimension 1 and 2. By using the characteristic  $p$  of the base field of the form  $2^f g - 1$  for a small odd integer  $g$ , our algorithm works by only 2-isogenies and  $(2, 2)$ -isogenies in the operations in  $\mathbb{F}_{p^2}$ . We apply our algorithm to SQIsign and compare the efficiency of the new algorithm with the existing one. Our analysis shows that the key generation and the signing in our algorithm are at least twice as fast as those in the existing algorithm at the NIST security level 1. This advantage becomes more significant at higher security levels. In addition, our algorithm also improves the efficiency of the verification in SQIsign.

**Keywords:** post-quantum cryptography · SQIsign · the Deuring correspondence · Kani's theorem

## 1 Introduction

Isogeny-based cryptography is a promising candidate for post-quantum cryptography. Many isogeny-based schemes use supersingular elliptic curves because the isogeny graph of supersingular elliptic curves has a more attractive structure than that of ordinary elliptic curves. Some of these schemes use the Deuring correspondence, which is a correspondence between supersingular elliptic curves and quaternion orders. SQIsign is a signature scheme proposed by De Feo, Kohel, Leroux, Petit and Wesolowski [12] that uses the Deuring correspondence. It was submitted to the additional digital signature candidates for the NIST post-quantum

cryptography standardization process [6]. In this paper, we refer the NIST submission of SQIsign as the SQISIGN to distinguish it from the name of the scheme. An advantage of the SQISIGN is that it has short key sizes and signature sizes compared to other candidates. Its disadvantage is that the signing algorithm is slow. This mainly comes from the computation of an isogeny corresponding to a quaternion ideal via the Deuring correspondence. We call an algorithm to compute an isogeny corresponding to a quaternion ideal an *ideal-to-isogeny algorithm*.

Ideal-to-isogeny algorithms are crucial for isogeny-based cryptography. Before SQIsign was proposed, ideal-to-isogeny algorithms appeared in the cryptanalysis by Eisenträger, Hallgren, Lauter, Morrison and Petit [20] and the signature scheme by Galbraith, Petit and Silva [23]. Although, the ideal-to-isogeny algorithms in these works have a polynomial-time complexity, they are not efficient in practice because they require operations on extension fields. The first efficient ideal-to-isogeny algorithm was proposed in SQIsign [12]. This algorithm does not require operations on extension fields, but it requires that the characteristic of the base field is in a special form. Later, the restriction on the characteristic was relaxed by De Feo, Leroux, Longa and Wesolowski [14].

Another important mathematical tool for isogeny-based cryptography is Kani's reducibility theorem [26]. This theorem gives a connection between isogenies between elliptic curves and isogenies between abelian surfaces, in other words, a connection between isogenies of dimension 1 and 2. Castryck and Decru [5] and Maino, Martindale, Panny, Pope, and Wesolowski [31] used this theorem to attack SIDH, which is an isogeny-based key exchange protocol by Jao and De Feo [25]. Robert [36] extended these attacks to attacks using a connection between isogenies of dimension 2 (resp. 4) and 4 (resp. 8). Later, this theorem was used to construct isogeny-based schemes, for example, a signature scheme by [10], a public-key encryption scheme by [2], a key encapsulation mechanism by [32], and an updatable public-key encryption scheme by [18].

Some of these schemes use the Deuring correspondence in addition to Kani's reducibility theorem. SQIsignHD [10] uses isogenies of dimension 4 or 8 to confirm the existence of an isogeny corresponding to a quaternion ideal in its verification algorithm. QFESTA [32] uses isogenies of dimension 2 and the Deuring correspondence to generate a random isogeny between elliptic curves of given degree. Ideal-to-isogeny algorithms using Kani's reducibility theorem have been proposed in verifiable random functions by [29] and in SILBE [18].

## 1.1 Our Contributions

Motivated by these developments, this paper advances this line of research by proposing a novel ideal-to-isogeny algorithm using isogenies of dimension 2. Our contributions are as follows:

1. Proposing a novel ideal-to-isogeny algorithm `IdealTolsogenyIQO`, which uses isogenies of dimension 2 and an embedding of an imaginary quadratic order into the endomorphism ring of the domain elliptic curve (IQO stands for Imaginary Quadratic Order).
2. Applying `IdealTolsogenyIQO` to SQIsign and comparing the efficiency of the new algorithm with the existing one.

Our algorithm is based on a similar idea as the algorithm in SILBE, which uses isogenies of dimension 4. Compared to the algorithm in SILBE, our algorithm has two advantages. The first advantage is using more efficient isogenies of dimension 2 instead of isogenies of dimension 4. The second advantage is that our algorithm does not require that the degree of the output isogeny of dimension 1 is prime to the degree of isogenies of dimension 2. Thanks to these advantages, we can use only 2-isogenies and  $(2, 2)$ -isogenies to run our algorithm in practice if we choose the characteristic of the base field properly.

As an application of our algorithm, we propose a new algorithm for SQIsign. Our algorithm uses the characteristics of the form  $2^f g - 1$  for a small odd integer  $g$ . The isogenies directly computed in our algorithm are only 2-isogenies and  $(2, 2)$ -isogenies. By using an efficient algorithm to compute  $(2, 2)$ -isogenies by Dartios, Maino, Pope, and Robert [11], we expect that the key generation and the signing in our algorithm are faster than those in the SQISIGN. The verification in our algorithm is faster than that in the SQISIGN because the number of the separations of the isogeny chain in the signature of our algorithm is smaller than that in the SQISIGN. Note that our algorithm does not affect the security of the SQISIGN, and the sizes of the keys and the signatures of our algorithm are almost the same as those of the SQISIGN because we just replace the ideal-to-isogeny algorithm in the SQISIGN.

## 1.2 Related Works

As mentioned in [29, §6], the ideal-to-isogeny algorithm in [29] could be applied to SQIsign in a manner similar to our algorithm. This algorithm also uses isogenies of dimension 2, but takes a different approach from our algorithm. We discuss a comparison with this algorithm in Sect. 4.6.

SQIsignHD [10] is a variant of SQIsign, which uses isogenies of dimension 4 or 8. The key generation and signing algorithms in SQIsignHD are more efficient than those in SQIsign while the verification algorithm in SQIsignHD is slower than that in SQIsign. In terms of key and signature sizes, SQIsignHD has the same key sizes as SQIsign and smaller signature sizes than SQIsign. Furthermore, SQIsignHD relies on distinct assumptions from SQIsign for security.

Although, we improve the efficiency of SQIsign, the key generation and signing algorithms in our algorithm are slower than those in SQIsignHD. Nonetheless, we contend that our approach remains valuable due to its fast verification process compared to these schemes. Furthermore, the exploration of diverse isogeny-based schemes based on distinct assumptions remains crucial.

At the same time as this work, other variants of SQIsign, SQIsign2D-West [1], SQIprime [17], and SQIsign2D-East [33], have been proposed. These variants use isogenies of dimension 2 or 4 and offer different trade-offs between efficiency and security. We leave the comparison with these schemes as future work. In addition, a new ideal-to-isogeny algorithm using isogenies of dimension 2 was proposed in SQIsign2D-West. It could be applied to SQIsign similarly to our algorithm. We also leave the comparison with this algorithm as future work.

### 1.3 Organization

The rest of this paper is organized as follows. In Sect. 2, we give the technical background on this paper. In particular, Sect. 2.1 gives the mathematical background, Sect. 2.2 gives the existing ideal-to-isogeny algorithms, and Sect. 2.3 explains the outline of SQIsign. In Sect. 3, we propose a novel ideal-to-isogeny algorithm using isogenies of dimension 2. In Sect. 4, we apply our algorithm to SQIsign and compare the efficiency of the new algorithm with the existing one. Finally, we conclude this paper in Sect. 6.

## 2 Preliminaries

This section gives the technical background on this paper. Throughout this paper, we let  $p$  be a prime number of cryptographic size, i.e.,  $p$  is at least about  $2^{256}$ .

### 2.1 Mathematical Background

In this subsection, we recall the mathematical background necessary for the rest of this paper.

**Supersingular Elliptic Curves.** Let  $E$  be an elliptic curve over a finite field of characteristic  $p$ . We denote the neutral element of  $E$  by  $O_E$ . For an integer  $n$ , the  $n$ -torsion subgroup of  $E$  is defined by  $E[n] = \{P \in E \mid nP = O_E\}$ . If  $E[p]$  is trivial, then  $E$  is called *supersingular*. A supersingular elliptic curve over a field of characteristic  $p$  is isomorphic to a curve  $E$  defined over  $\mathbb{F}_{p^2}$  such that the  $p^2$ -th power Frobenius endomorphism of  $E$  is the multiplication-by- $(-p)$  map. Then we have  $E(\mathbb{F}_{p^2}) = E[p + 1]$ . This property is preserved under isogenies over  $\mathbb{F}_{p^2}$ , i.e., if there exists an isogeny  $E \rightarrow E'$  defined over  $\mathbb{F}_{p^2}$  then  $E'$  is also a supersingular elliptic curve such that  $E'(\mathbb{F}_{p^2}) = E'[p + 1]$ . In the rest of this paper, we assume that all elliptic curves are supersingular and satisfy the above property.

**Abelian Surfaces.** An elliptic curve is an abelian variety of dimension 1. The generalization of elliptic curves to dimension 2 is called an *abelian surface*. An abelian surface is *principally polarized* if it is isomorphic to its dual abelian surface. A principally polarized abelian surface is isomorphic (over an algebraically closed field) to the Jacobian of a genus-2 hyperelliptic curve or the product of two elliptic curves.

**Isogenies.** An *isogeny* is a rational map between principally polarized abelian varieties which is a surjective group homomorphism and has finite kernel. The *degree* of an isogeny  $\varphi$  is its degree as a rational map and denoted by  $\deg \varphi$ . An isogeny  $\varphi$  is *separable* if  $\#\ker \varphi = \deg \varphi$ . A separable isogeny is uniquely

determined by its kernel up to post-composition of isomorphism. For an isogeny  $\varphi : A \rightarrow B$ , the *dual isogeny* of  $\varphi$  is the isogeny  $\hat{\varphi} : B \rightarrow A$  such that  $\hat{\varphi} \circ \varphi$  is equal to the multiplication-by-deg  $\varphi$  map on  $A$ . Note that the dual isogeny uniquely exists.

Let  $\ell$  be a positive integer. We say an isogeny  $\varphi$  between two elliptic curves is an  $\ell$ -isogeny if  $\ker \varphi$  is a cyclic group of order  $\ell$ . We say an isogeny  $\varphi$  between two principally polarized abelian surfaces is an  $(\ell, \ell)$ -isogeny if the Weil pairing acts trivially on  $\ker \varphi$  and the order of  $\ker \varphi$  is  $\ell^2$ .

**Algorithms to Compute Isogenies.** An isogeny between elliptic curves can be computed by Vélu’s formulas [39]. Let  $\varphi : E \rightarrow E'$  be an  $\ell$ -isogeny between elliptic curves. Vélu’s formulas give an algorithm to compute  $E'$  with input  $E$  and a generator of  $\ker \varphi$  in  $O(\ell)$  operations on a field where the generator is defined. For an additional input  $P \in E$ , we can compute  $\varphi(P)$  in  $O(\ell)$  operations on a field where the generator and  $P$  are defined. These computational costs were improved to  $\tilde{O}(\sqrt{\ell})$  by [3].

Algorithms to compute a  $(2, 2)$ -isogeny can be found in [38] and [24]. Recently, an efficient algorithm for a  $(2, 2)$ -isogeny using theta functions was given by [11]. An algorithm for a general degree was given by [8] and later improved by [30]. The computational cost of this algorithm is  $O(\ell^2)$  operations on a field where a generator of the kernel is defined.

Let  $d$  be a positive integer prime to  $p$  having the prime factorization  $d = \prod_i \ell_i$  and  $\varphi$  be a  $d$ -isogeny or  $(d, d)$ -isogeny. Then we can compute  $\varphi$  as the composition of  $\ell_i$ -isogenies or  $(\ell_i, \ell_i)$ -isogenies. Therefore, if  $d$  is smooth and a generator of  $\ker \varphi$  are defined over a  $\mathbb{F}_{p^k}$  of  $k \in \text{poly}(\log p)$ , then we can compute a  $d$ -isogeny in polynomial time in  $\log p$ .

**Quaternion Algebras.** We denote by  $\mathcal{B}_{p,\infty}$  the quaternion algebra over  $\mathbb{Q}$  ramified at  $p$  and  $\infty$ . The quaternion algebra  $\mathcal{B}_{p,\infty}$  has  $\mathbb{Q}$ -basis  $\{1, \mathbf{i}, \mathbf{j}, \mathbf{k}\}$  such that  $\mathbf{i}^2 = -q, \mathbf{j}^2 = -p, \mathbf{k} = \mathbf{ij} = -\mathbf{ji}$  for some positive integer  $q$ . If  $p \equiv 3 \pmod{4}$ , then we let  $q = 1$ . If  $p \equiv 5 \pmod{8}$ , then we let  $q = 2$ . Otherwise, we let  $q$  be the smallest prime number such that  $q$  is a quadratic non-residue modulo  $p$ . The *canonical involution* on  $\mathcal{B}_{p,\infty}$  is defined by  $\alpha = a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k} \mapsto \bar{\alpha} := a - b\mathbf{i} - c\mathbf{j} - d\mathbf{k}$ . The *trace* and the *norm* of  $\alpha$  are defined by  $\text{tr}(\alpha) := \alpha + \bar{\alpha}$  and  $\text{n}(\alpha) := \alpha\bar{\alpha}$ , respectively. An order in  $\mathcal{B}_{p,\infty}$  is a subring of  $\mathcal{B}_{p,\infty}$  that is a free  $\mathbb{Z}$ -module of rank 4. A maximal order in  $\mathcal{B}_{p,\infty}$  is an order that is maximal with respect to inclusion. A *fractional ideal* of  $\mathcal{B}_{p,\infty}$  is a  $\mathbb{Z}$ -submodule of  $\mathcal{B}_{p,\infty}$  of rank 4. Let  $I$  be a fractional ideal of  $\mathcal{B}_{p,\infty}$ . We define the fractional ideal  $\bar{I} := \{\bar{\alpha} \mid \alpha \in I\}$ . The *left order* of  $I$  is defined by  $\mathcal{O}_L(I) := \{x \in \mathcal{B}_{p,\infty} \mid xI \subset I\}$  and we define the *right order*  $\mathcal{O}_R(I)$  of  $I$  similarly. For an order  $\mathcal{O}$  of  $\mathcal{B}_{p,\infty}$ , we say  $I$  is a *left (or right)  $\mathcal{O}$ -ideal* if  $I$  is a left (or right) ideal of  $\mathcal{O}$  in the usual sense. If  $I$  is a left  $\mathcal{O}$ -ideal for a maximal order  $\mathcal{O}$ , then  $\mathcal{O}_L(I) = \mathcal{O}$  and  $\mathcal{O}_R(I)$  is a maximal order. If  $I$  is contained in an order of  $\mathcal{B}_{p,\infty}$ , then we define the *norm* of  $I$  by  $\text{n}(I) := \text{gcd}(\{\text{n}(\alpha) \mid \alpha \in I\})$ . For  $\alpha \in I$ , we define the *normalized norm of  $\alpha$*  by

$q_I(\alpha) := n(\alpha)/n(I)$ . By the definition of the norm of  $I$ , the normalized norm  $q_I(\alpha)$  is an integer for all  $\alpha \in I$ .

**The Deuring Correspondence.** Deuring [16] showed that the endomorphism ring of a supersingular elliptic curve over  $\mathbb{F}_{p^2}$  is isomorphic to a maximal order of  $\mathcal{B}_{p,\infty}$  and gave a correspondence (*Deuring correspondence*) where a supersingular elliptic  $E$  curve over  $\mathbb{F}_{p^2}$  corresponds to a maximal order isomorphic to  $\text{End}(E)$ .

Fix a supersingular elliptic curve  $E_0$  and an isomorphism  $\iota : \mathcal{O}_0 \rightarrow \text{End}(E_0)$ . For a left  $\mathcal{O}_0$ -ideal  $I$ , we define the  $I$ -torsion subgroup of  $E_0$  by  $E_0[I] = \{P \in E_0 \mid \iota(\alpha)(P) = 0 \text{ for all } \alpha \in I\}$ . If  $n(I)$  is not divisible by  $p$ , then  $E_0[I]$  is a subgroup of  $E_0$  of order  $n(I)$ . In this case, we define an isogeny corresponding to  $I$  by an isogeny with kernel  $E_0[I]$  and denote it by  $\varphi_I$ . Let  $E$  be the codomain of  $\varphi_I$ . Then  $\text{End}(E)$  is isomorphic to  $\mathcal{O}_R(I)$ . In particular, an isomorphism is induced by

$$\mathcal{B}_{p,\infty} \rightarrow \text{End}(E) \otimes_{\mathbb{Z}} \mathbb{Q}; \alpha \mapsto \frac{1}{n(I)}\varphi_I \circ \iota(\alpha) \circ \hat{\varphi}_I. \tag{1}$$

The fraction ideal  $\bar{I}$  corresponds to the dual isogeny  $\hat{\varphi}_I$ . Let  $J$  be a left  $\mathcal{O}_R(I)$ -ideal and  $\varphi_J$  be an isogeny corresponding to  $J$  via the above isomorphism. Then the composition  $\varphi_J \circ \varphi_I$  is an isogeny corresponding to  $IJ$ . If  $n(I)$  is prime to  $n(J)$ , then  $\ker \varphi_J = \varphi_I(E_0[IJ] \cap E_0[n(J)])$ . For more detailed discussion for the relation between ideals and isogenies, see [12, §4].

For left  $\mathcal{O}_0$ -ideals  $I_1$  and  $I_2$ , the codomains of  $\varphi_{I_1}$  and  $\varphi_{I_2}$  are isomorphic if and only if there exists  $\alpha \in \mathcal{B}_{p,\infty}$  such that  $I_1 = I_2\alpha$ . If this is the case, we say  $I_1$  and  $I_2$  are *equivalent* and denote it by  $I_1 \sim I_2$ .

**Special Extremal Orders.** Let  $R$  be the integer ring of  $\mathbb{Q}(\mathbf{i})$ . We say a maximal order  $\mathcal{O}$  in  $\mathcal{B}_{p,\infty}$  is *special extremal* if  $\mathcal{O}$  contains  $R + \mathbf{j}R$ . In this paper, we mainly focus on the case  $p \equiv 3 \pmod{4}$ . In this case, the maximal order  $\mathcal{O}_0 := \left\langle 1, \mathbf{i}, \frac{1+\mathbf{j}}{2}, \frac{1+\mathbf{k}}{2} \right\rangle_{\mathbb{Z}}$  is a special extremal order and the supersingular elliptic curve with  $j$ -invariant 1728 corresponds to  $\mathcal{O}_0$  via the Deuring correspondence. Let  $E_0$  be the supersingular elliptic curve over  $\mathbb{F}_{p^2}$  defined by  $y^2 = x^3 + x$ , which has  $j$ -invariant 1728. Then an isomorphism  $\mathcal{O}_0 \rightarrow \text{End}(E_0)$  is induced by  $\mathbf{i} \mapsto ((x, y) \mapsto (-x, \sqrt{-1}y))$  and mapping  $\mathbf{j}$  to the  $p$ -th power Frobenius endomorphism.

**KLPT Algorithms.** An algorithm to transform an ideal to another equivalent ideal is given by [27], which is called the *KLPT algorithm*. Let  $\mathcal{O}_0$  be a special extremal order in  $\mathcal{B}_{p,\infty}$ . The KLPT algorithm takes a left  $\mathcal{O}_0$ -ideal  $I$  and a smooth integer  $n > p^{3.5}$  as input and outputs a left  $\mathcal{O}_0$ -ideal  $J$  such that  $J \sim I$  and  $n(J) = n$ . Its computational cost is bounded by a polynomial in  $\log p$  under heuristic assumptions. Later, the bound  $p^{3.5}$  was improved to  $p^3$  by [35].

The KLPT algorithm was generalized to ideals of arbitrary maximal orders by [12]. We call this the *generalized KLPT algorithm*. Let  $\mathcal{O}$  be a maximal order of  $\mathcal{B}_{p,\infty}$  and  $I_0$  be a left  $\mathcal{O}_0$ -ideal such that  $\mathcal{O}_R(I_0) = \mathcal{O}$ . The generalized KLPT

algorithm takes  $I_0$ , a left  $\mathcal{O}$ -ideal  $I$ , and a smooth integer  $n > p^3 n(I_0)^3$  as input and outputs a left  $\mathcal{O}$ -ideal  $J$  such that  $J \sim I$  and  $n(J) = n$ .

These algorithms are based on the fact that  $I\bar{\alpha}/n(I)$  is a left  $\mathcal{O}_L(I)$ -ideal of norm  $q_I(\alpha)$  for  $\alpha \in I$  [27, Lemma 5]. Indeed, these algorithms find  $\alpha \in I$  such that  $q_I(\alpha) = n$  and output  $I\bar{\alpha}/n(I)$ . We denote the ideal  $I\bar{\alpha}/n(I)$  by  $\chi_I(\alpha)$ . Then  $\hat{\varphi}_{\chi_I(\alpha)} \circ \varphi_I$  is equal to  $\alpha$  as endomorphisms up to post-composition of an automorphism of  $E_0$ . Note that the right order of  $\chi_I(\alpha)$  is  $\alpha \mathcal{O}_R(I) \alpha^{-1}$ , which is isomorphic to  $\mathcal{O}_R(I)$  but not equal to  $\mathcal{O}_R(I)$  in general.

**Kani’s Reducibility Theorem.** Let  $d_1$  and  $d_2$  be positive integers prime to each other and  $p$ . Let  $\varphi_1 : E_0 \rightarrow E_1$  be a  $d_1$ -isogeny and  $\varphi_2 : E_0 \rightarrow E_2$  be a  $d_2$ -isogeny between elliptic curves over a field of characteristic  $p$ . Then we say an isogeny with kernel  $\varphi_1(\ker \varphi_2)$  a *push-forward of  $\varphi_2$  by  $\varphi_1$*  and denote it by  $\varphi_{1*}\varphi_2$ . Since  $\ker((\varphi_{1*}\varphi_2) \circ \varphi_1) = \langle \ker \varphi_1, \ker \varphi_2 \rangle = \ker((\varphi_{2*}\varphi_1) \circ \varphi_2)$ , the codomains of  $\varphi_{1*}\varphi_2$  and  $\varphi_{2*}\varphi_1$  are isomorphic. Let  $F$  be the codomain of  $\varphi_{1*}\varphi_2$ . Then we can take  $\varphi_{2*}\varphi_1$  so that the following diagram commutes:

$$\begin{array}{ccc} E_0 & \xrightarrow{\varphi_1} & E_1 \\ \varphi_2 \downarrow & & \downarrow \varphi_{1*}\varphi_2 \\ E_2 & \xrightarrow{\varphi_{2*}\varphi_1} & F. \end{array}$$

Kani [26] showed that this diagram induces an isogeny  $E_1 \times E_2 \rightarrow E_0 \times F$ . More precisely, we have the following theorem based on Kani’s reducibility theorem [26, Theorem 2.3].

**Theorem 1 ([31, Theorem 1]).** *We use the same notation as above and let  $d = d_1 + d_2$ . Suppose that we take the push-forwards so that the above diagram is commutative. We define an isogeny*

$$\Phi = \begin{pmatrix} \hat{\varphi}_1 & \hat{\varphi}_2 \\ -\varphi_{1*}\varphi_2 & \varphi_{2*}\varphi_1 \end{pmatrix} : E_1 \times E_2 \rightarrow E_0 \times F,$$

*i.e.,  $\Phi((P_1, P_2)) = (\hat{\varphi}_1(P_1) + \hat{\varphi}_2(P_2), -\varphi_{1*}\varphi_2(P_1) + \varphi_{2*}\varphi_1(P_2))$  for  $P_1 \in E_1$  and  $P_2 \in E_2$ . Then  $\Phi$  is a  $(d, d)$ -isogeny with kernel  $\{(\varphi_1(P), \varphi_2(P)) \mid P \in E_0[d]\}$ .*

This theorem says that we can compute the images of any points under  $\hat{\varphi}_1$  and  $\hat{\varphi}_2$  by using the images of a basis of  $E_0[d]$  under  $\varphi_1$  and  $\varphi_2$ .

If  $F$  is not isomorphic to  $E_0$ , then an isogeny with the same kernel as  $\Phi$  is of the form  $\begin{pmatrix} \iota_0 & 0 \\ 0 & \iota \end{pmatrix} \circ \Phi$  or  $\begin{pmatrix} 0 & \iota \\ \iota_0 & 0 \end{pmatrix} \circ \Phi$ , where  $\iota_0$  and  $\iota$  are automorphisms of  $E_0$  and  $F$ , respectively. In this case, we can compute the images of any point in  $E_1$  under  $\hat{\varphi}_1$  by Algorithm 1. In the output of Algorithm 1, the automorphism  $\iota_0$  is not determined by the input and depends on the choice of  $\Phi$ .



---

**Algorithm 1: EvalByKani**

---

**Input:**  $d, E_0, E_1, E_2$  in Theorem 1,  $\varphi_1(P), \varphi_2(P), \varphi_1(Q), \varphi_2(Q)$ , where  $(P, Q)$  is a basis of  $E_0[d]$ , and a finite subset  $S \subset E_1(\mathbb{F}_{p^2})$ .

**Output:** the image of  $S$  under  $\iota_0 \circ \hat{\varphi}_1$ , where  $\iota_0$  is an automorphism of  $E_0$ .

- 1 Let  $\Phi$  be a  $(d, d)$ -isogeny with kernel  $\langle (\varphi_1(P), \varphi_2(P)), (\varphi_1(Q), \varphi_2(Q)) \rangle$ ;
  - 2 Let  $\text{pr}_{E_0}$  be the projection to  $E_0$  from the codomain of  $\Phi$ ;
  - 3 **return**  $\{\text{pr}_{E_0} \circ \Phi((R, 0_{E_2})) \mid R \in S\}$ ;
- 

## 2.2 Ideal-to-Isogeny Algorithms

In this subsection, we recall some of the existing algorithms for computing the codomain of an isogeny corresponding to a given ideal.

Let  $E_0$  be a supersingular elliptic curve over  $\mathbb{F}_{p^2}$  whose endomorphism ring is isomorphic to a special extremal order  $\mathcal{O}_0$  in  $\mathcal{B}_{p,\infty}$ , and  $\iota : \mathcal{O}_0 \rightarrow \text{End}(E_0)$  be an isomorphism. Suppose that we can efficiently compute  $\iota(\alpha)(P)$  for  $\alpha \in \mathcal{O}_0$  and  $P \in E_0(\mathbb{F}_{p^2})$ . Given a left  $\mathcal{O}_0$ -ideal  $I$ , we consider an algorithm to compute the codomain of an isogeny corresponding to  $I$ . We call this algorithm an *ideal-to-isogeny algorithm*.

By using the KLPT algorithm, we can use an ideal  $J$  such that  $J \sim I$  and  $n(J)$  is smooth and greater than  $p^3$  instead of  $I$ . Since  $\varphi_J = n(J)$  is smooth, we can compute  $\varphi_J$  in polynomial time if  $E[J]$  is defined over a field of polynomial size. However,  $E[J]$  is in an exponential-size field in general. The following algorithms deal with this issue.

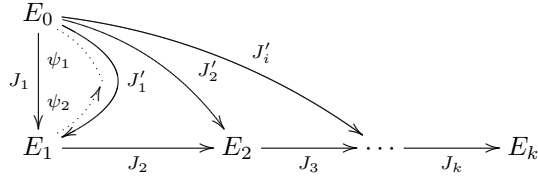
**Algorithm for Power-Smooth Norms.** An ideal-to-isogeny algorithm was first proposed by [23]. Their algorithm uses an ideal  $J$  such that  $J \sim I$  and  $n(J)$  is power-smooth, i.e., any prime power dividing  $n(J)$  is small. Its computation requires operations on extension fields of  $\mathbb{F}_{p^2}$ , thus the algorithm is not efficient in practice.

**Algorithm in the Original SQIsign.** An ideal-to-isogeny algorithm that works in  $\mathbb{F}_{p^2}$  was proposed by [12]. This algorithm requires  $p$  to be of a special form while the algorithm of [23] does not. In particular, the algorithm by [12] requires that  $p^2 - 1$  is divisible by  $\ell^f$  and  $T$ , where  $\ell$  is a small prime number,  $f$  is an integer and  $T$  is a smooth integer greater than  $p^{1.5}$  and prime to  $\ell$ . In this setting, we can compute  $\ell^f$ -isogenies and  $T$ -isogenies over  $\mathbb{F}_{p^2}$  by using the method in [9]. The basic idea of the algorithm is as follows:

1. Compute a left  $\mathcal{O}_0$ -ideal  $J$  such that  $J \sim I$  and  $n(J)$  is a power of  $\ell$ .
2. Divide  $J$  into the product  $J_1 \cdots J_k$  such that  $n(J_i) = \ell^f$  (for simplicity, we assume  $n(J) = \ell^{kf}$  for an integer  $k$ ).
3. Let  $J'_0 = \mathcal{O}_0$  and  $\alpha = 1$ .
4. For  $i = 1, \dots, k$ :
  - (a) Compute  $\varphi_{J_i} : E_{i-1} \rightarrow E_i$  by  $\ker \varphi_{J_i} = \varphi_{J'_{i-1}}(E_0[J'_{i-1}\alpha J_i\alpha^{-1}] \cap E_0[\ell^f])$ .

- (b) Find  $\alpha \in J_1 \cdots J_i$  such that  $q_{J_1 \dots J_i}(\alpha) = T^2$  by using the KLPT algorithm.
- (c) Let  $J'_i = \chi_{J_1 \dots J_i}(\alpha)$ .
- (d) Compute isogenies  $\psi_1$  with kernel  $E_0[J'_i] \cap E_0[T]$  and  $\psi_2$  with kernel  $\varphi_{J_i} \circ \cdots \circ \varphi_{J_1}(E_0[\alpha] \cap E_0[T])$ .
- (e) Obtain  $\varphi_{J'_i} = \hat{\psi}_2 \circ \psi_1$ .

The following diagram illustrates the above algorithm:



Note that  $\alpha J_i \alpha^{-1}$  in Step (4.a) is a left  $\mathcal{O}_R(J'_{i-1})$ -ideal corresponding to  $J_i$ . We also note that Step (4.d) and (4.e) are based on the fact that  $\alpha = \hat{\varphi}_{J'_i} \circ \varphi_{J_i} \circ \cdots \circ \varphi_{J_1}$  up to post-composition of an automorphism of  $E_0$ . The algorithm by [12] is a further elaboration of the above idea. See [12, §8.1] for more details.

**IdealToIsogenyEichler.** The restriction on  $p$  in the above algorithm was relaxed by [14]. In particular, the lower bound on  $T$  in the restriction was improved to  $p^{1.25}$ . They use an endomorphism of each  $E_i$  instead of the isogeny  $\varphi_{J'_i}$ . The rough idea of their algorithm is as follows:

We use the same notation as above and consider the computation of  $\varphi_{J_{i+1}}$  from  $E_i$  and  $\varphi_{J_i}$ . Let  $\mathcal{O}$  be an order isomorphic to  $\text{End}(E_i)$  and  $\beta \in \mathcal{O}$  such that  $J_{i+1} = \mathcal{O}\beta + \mathcal{O}\ell^f$ . Then we search  $\theta \in \mathcal{O}$  and coprime integers  $C, D$  such that  $n(\theta) = T^2$  and  $\beta(C + D\theta) \in \bar{J}_i$ . The latter condition means that  $(C + D\theta)(P)$  generates  $E_i[J_{i+1}]$  for a generator  $P$  of  $\ker \hat{\varphi}_{J_i}$ . From this, we can compute  $\varphi_{J_{i+1}}$  by using Vélú’s formulas.

Algorithm 4 in [14] shows how to find  $\theta$  and  $C, D$ . This algorithm succeeds if  $n(\theta) > p^{2.5}$ , thus we can take  $T > p^{1.25}$ .

We call this algorithm **IdealTolsogenyEichler**. The **SQISIGN** uses this algorithm as an ideal-to-isogeny algorithm.

**Algorithm in DeuringVRF.** DeuringVRF is the family of verifiable random functions proposed by [29]. To construct this scheme, an ideal-to-isogeny algorithm was proposed. This algorithm can be seen as an extension of **IdealTolsogenyEichler** using isogenies of dimension 2. In this algorithm, the image of  $P$  under the endomorphism  $\theta$  in the explanation above is computed by using a 2-dimensional isogeny  $E_i \times E_i \rightarrow E_i \times E_i$ . See Algorithm 7 in [29] for more details.

**Algorithm in the Key Generation in SILBE.** An ideal-to-isogeny algorithm using higher-dimension isogenies was proposed by [18], which is used in the key generation in an updatable public key encryption scheme SILBE. This algorithm is based on the similar idea as in the algorithm in the original SQIsign. To compute the auxiliary isogenies  $\varphi_{J_i}$ , it uses an extension of Kani’s reducibility theorem to dimension 4 by [36]. See [18, §3.1] for more details.

### 2.3 SQIsign

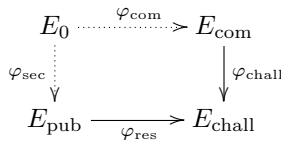
SQIsign is a signature scheme proposed by [12], which uses the generalized KLPT algorithm and an ideal-to-isogeny algorithm as building blocks.

**Overview.** Let  $\mathcal{O}_0$  be a special extremal order and  $E_0$  a supersingular elliptic curve over  $\mathbb{F}_{p^2}$  whose endomorphism ring is isomorphic to  $\mathcal{O}_0$ . We consider the following zero-knowledge proof.

The public parameters are  $p$ ,  $\mathcal{O}_0$ , and  $E_0$ . The protocol proves the knowledge of a secret left  $\mathcal{O}_0$ -ideal  $I_{\text{sec}}$  with a public key  $E_{\text{pub}}$ , which is the codomain of an isogeny  $\varphi_{\text{sec}}$  corresponding to  $I_{\text{sec}}$ . The protocol is as follows:

1. The prover computes an isogeny  $\varphi_{\text{com}} : E_0 \rightarrow E_{\text{com}}$  and sends  $E_{\text{com}}$  as a commitment to the verifier.
2. The verifier computes an isogeny  $\varphi_{\text{chall}} : E_{\text{com}} \rightarrow E_{\text{chall}}$  and sends  $\varphi_{\text{chall}}$  and  $E_{\text{chall}}$  as a challenge to the prover.
3. The prover computes ideals  $I_{\text{com}}$  corresponding to  $\varphi_{\text{com}}$  and  $I_{\text{chall}}$  corresponding to  $\varphi_{\text{chall}}$ .
4. The prover applies the generalized KLPT algorithm to  $I_{\text{sec}}$  and  $\bar{I}_{\text{sec}}I_{\text{com}}I_{\text{chall}}$  and computes an ideal  $I_{\text{res}} \sim \bar{I}_{\text{sec}}I_{\text{com}}I_{\text{chall}}$ .
5. The prover computes  $\varphi_{\text{res}}$  corresponding to  $I_{\text{res}}$  and sends it to the verifier.
6. The verifier check that  $\varphi_{\text{res}}$  is an isogeny from  $E_{\text{pub}}$  to  $E_{\text{chall}}$  and the kernel of  $\hat{\varphi}_{\text{chall}} \circ \varphi_{\text{res}}$  is cyclic.

The following diagram illustrates the above protocol, where the dotted arrows represent the isogenies kept secret by the prover:



SQIsign is a signature scheme obtained by applying the Fiat-Shamir transform [21] to the above protocol.

In the following, we describe the each step of the SQISIGN in more detail.

**Parameter.** As we mentioned in Sect. 2.2, the SQISIGN uses IdealTolsogenyEichler as an ideal-to-isogeny algorithm. For efficiency, we use 2 as  $\ell$ . Therefore, we use  $p$  such that  $p^2 - 1$  is divisible by  $2^f$  and  $T$  for an odd smooth integer  $T$  greater than  $p^{1.25}$ .

Let  $\lambda$  be a security parameter. To achieve a  $\lambda$ -bit security level, we require the following conditions:

- $p \approx 2^{2\lambda}$  (to address the attacks by [15] and [19]),
- $\deg \varphi_{\text{com}} \approx 2^{2\lambda}$  (to address the meet-in-the-middle attack [25, §5.2]),
- $\deg \varphi_{\text{chall}} \approx 2^\lambda$  (to ensure the challenge space of size  $2^\lambda$ ).

In the SQISIGN, the prime  $p$  satisfies that  $T$  is divisible by  $3^g$  such that  $2^f 3^g \approx 2^\lambda$  and  $T/3^g$  is prime to 3 and greater than  $2^\lambda$ . And we set  $\deg \varphi_{\text{com}} = T/3^g$  and  $\deg \varphi_{\text{chall}} = 2^f 3^g$ .

**Key Generation.** The key generation algorithm is as follows:

1. Sample a prime  $D_{\text{sec}} < p^{1/4}$  such that  $D_{\text{sec}} \equiv 3 \pmod{4}$  uniformly at random.
2. Sample a left  $\mathcal{O}_0$ -ideal  $I_{\text{sec}}$  of norm  $D_{\text{sec}}$  uniformly at random.
3. Compute  $J$  be a left  $\mathcal{O}_0$ -ideal such that  $J \sim I_{\text{sec}}$  and  $n(J)$  is a power of 2 by the KLPT algorithm.
4. Compute the codomain  $E_{\text{pub}}$  of an isogeny  $\varphi_{\text{sec}}$  corresponding to  $I_{\text{sec}}$  by IdealTolsogenyEichler.
5. Output a public key  $E_{\text{pub}}$  and a secret key  $I_{\text{sec}}$ .

The reason that we take  $D_{\text{sec}} < p^{1/4}$  is to reduce the norm of the output of the generalized KLPT algorithm. The bound  $p^{1/4}$  is the minimum to make the size of the secret key space larger than  $2^\lambda$ .

**Commitment.** A commitment is computed by using Vélu’s formulas. Taking a point  $K$  of order  $T/3^g$  on  $E_0$ , we compute an isogeny  $\varphi_{\text{com}}$  with kernel  $\langle K \rangle$ . Then we output the codomain  $E_{\text{com}}$  of  $\varphi_{\text{com}}$  as a commitment.

**Challenge.** A challenge  $c$  is sampled from the integers in  $[0, 2^f 3^g)$ . Then the corresponding isogeny  $\varphi_{\text{chall}}$  is computed by using Vélu’s formulas from the kernel  $\langle P_{E_{\text{com}}} + cQ_{E_{\text{com}}} \rangle$ , where  $(P_{E_{\text{com}}}, Q_{E_{\text{com}}})$  is a deterministic basis of  $E_{\text{com}}[2^f 3^g]$ .

**Response.** Let  $\mathcal{O}$  be the right order of  $I_{\text{sec}}$ . Since  $n(I_{\text{sec}}) < p^{1/4}$ , we can find a left  $\mathcal{O}$ -ideal of norm approximately  $p^{3.75}$  equivalent to a given left  $\mathcal{O}_0$ -ideal by the generalized KLPT algorithm. Let  $k$  be an integer such that  $2^{kf} \approx p^{3.75}$ . The degree of  $\varphi_{\text{res}}$  is set to  $2^{kf}$ .

Given  $E_{\text{pub}}, E_{\text{com}}, I_{\text{sec}}, \varphi_{\text{com}}, \varphi_{\text{chall}}$ , the corresponding response is computed as follows:

1. Compute the left  $\mathcal{O}_0$ -ideal  $I_{\text{com}}$  corresponding to  $\varphi_{\text{com}}$ .

2. Compute the left  $\mathcal{O}_0$ -ideal  $I'_{\text{chall}}$  corresponding to an isogeny with kernel  $\hat{\varphi}_{\text{com}}(P_{E_{\text{com}}} + cQ_{E_{\text{com}}})$ .
3. Compute a left  $\mathcal{O}_R(I_{\text{sec}})$ -ideal  $I_{\text{res}}$  such that  $I_{\text{res}} \sim \bar{I}_{\text{sec}}(I_{\text{com}} \cap I'_{\text{chall}})$  and  $n(I_{\text{res}}) = 2^{kf}$  by the generalized KLPT algorithm.
4. Compute an isogeny  $\varphi_{\text{res}} = \varphi_{\text{res},k} \circ \cdots \circ \varphi_{\text{res},1}$  corresponding to  $I_{\text{res}}$  by **Ideal-TolsogenyEichler**, where  $\deg \varphi_{\text{res},i} = 2^f$  for  $i = 1, \dots, k$ .
5. Output the sequence of generators of  $\ker \varphi_{\text{res},1}, \dots, \ker \varphi_{\text{res},k}$  as a response.

**Verification.** The verification checks the following:

1. The codomain of the composition  $\varphi_{\text{res},k} \circ \cdots \circ \varphi_{\text{res},1}$  is isomorphic to  $E_{\text{chall}}$ .
2. The kernel of  $\hat{\varphi}_{\text{chall}} \circ \varphi_{\text{res},k} \circ \cdots \circ \varphi_{\text{res},1}$  is cyclic.

**Compression.** To reduce the size of the response, a generator of the kernel of  $\varphi_{\text{res},i}$  is represented by coefficients of the linear combination of a deterministic basis of the  $2^f$ -torsion subgroup of the domain. The detail is as follows: Let  $E_i$  be the domain of  $\varphi_{\text{res},i}$  and  $(P_{E_i}, Q_{E_i})$  be a basis of  $E_i[2^f]$  that is computed deterministically. Then  $\ker \varphi_{\text{res},i}$  is generated by a point of the form  $aP_{E_i} + Q_{E_i}$  or  $P_{E_i} + aQ_{E_i}$  for an integer  $a \in [0, 2^f)$ . Therefore, we can represent a generator of  $\ker \varphi_{\text{res},i}$  by the integer  $a$  and a bit indicating the form of the generator.

Since the **SQISIGN** is the signature scheme obtained by applying the Fiat-Shamir transform to the above protocol, a signature of the **SQISIGN** is a pair of a commitment and a response. In the **SQISIGN**, a commitment is compressed by generators of the kernels of  $\varphi_{\text{chall}}$  and its dual isogeny and these generators are compressed by the above method. For details, see §3.4 and §3.5 in the document in [6].

### 3 New Algorithms

In this section, we give new ideal-to-isogeny algorithms. Our algorithms are based on the same idea as in the algorithm in the original **SQISign** and the algorithm in the key generation in **SILBE** explained in Sect. 2.2. In particular, we use isogenies of dimension 2 instead of  $T$ -isogenies or isogenies of dimension 4.

To use isogenies of dimension 2, we use an embedding of an imaginary quadratic order into the endomorphism ring of the domain curve. Unlike the algorithm in **SILBE**, our algorithms allow that the degrees of isogenies of dimension 1 and 2 have the same prime factors. This flexibility enables the use of 2-isogenies and (2, 2)-isogenies, thereby enhancing algorithmic efficiency.

#### 3.1 Setting

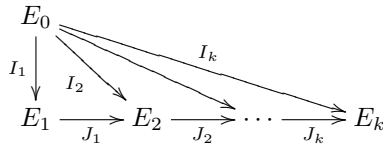
Let  $m_1$  and  $m_2$  be smooth integers such that  $p = m_1 m_2 f - 1$  is a prime number for a small positive integer  $f$ . We assume that  $m_2 > \sqrt{p}$ . Note that we do not require  $m_1$  and  $m_2$  to be coprime. Let  $\mathcal{O}_0$  be a special extremal order in

$\mathcal{B}_{p,\infty}$ ,  $R$  be the integer ring of  $\mathbb{Q}(\mathbf{i})$  contained in  $\mathcal{O}_0$ , and  $E_0$  be a supersingular elliptic curve over  $\mathbb{F}_{p^2}$  whose endomorphism ring is isomorphic to  $\mathcal{O}_0$ . We fix an isomorphism  $\iota : \mathcal{O}_0 \rightarrow \text{End}(E_0)$  and identify  $\mathcal{O}_0$  with  $\text{End}(E_0)$  by  $\iota$ . We also fix a basis  $(P_0, Q_0)$  of  $E_0[m_1m_2]$ . We assume that these parameters are implicitly given as input for the algorithms in this section.

Let  $I_1$  be a left  $\mathcal{O}_0$ -ideal such that  $n(I_1)$  is prime to  $m_1m_2$ ,  $\mathcal{O}$  be the right order of  $I_1$ , and  $E$  be the codomain of  $\varphi_{I_1}$ . Suppose that we know the images  $\varphi_{I_1}(P_0)$  and  $\varphi_{I_1}(Q_0)$ . Given a left  $\mathcal{O}$ -ideal  $J$ , we consider an algorithm to compute the codomain of  $\varphi_J$ . Here, the correspondence between  $J$  and  $\varphi_J$  is given by the isomorphism induced by (1). By using generalized KLPT algorithm, we can assume that  $n(J)$  is a power of  $m_1$ .

### 3.2 Core of the Algorithm

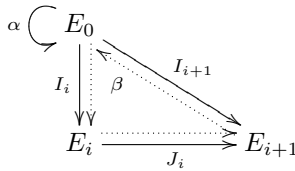
Decompose  $J$  into the product  $J = J_1 \cdots J_k$  such that  $n(J_i) = m_1$ . For each  $i$ , the  $J_i$ -torsion subgroup is rational over  $\mathbb{F}_{p^2}$ . Therefore, we can compute an isogeny corresponding to  $J_i$  by Vélu’s formulas if we know a generator of its kernel. For  $J_1$ , we have  $\ker \varphi_{J_1} = \varphi_{I_1}(E_0[I_1J_1] \cap E_0[m_1])$  since  $\varphi_{J_1} \circ \varphi_{I_0}$  corresponds to  $I_1J_1$  and  $n(I_1)$  is prime to  $m_1$ . Therefore, we can compute the codomain of  $\varphi_{J_1}$  by using  $\varphi_{I_1}(P_0)$  and  $\varphi_{I_1}(Q_0)$ . Similarly, we can compute the codomain of  $\varphi_{J_i}$  for  $i > 1$  if we have an ideal  $I_i$  such that  $I_i \sim I_1J_1 \cdots J_{i-1}$  and  $n(I_i)$  is prime to  $m_1$ , and the images of a basis of  $E_0[m_1]$  under  $\varphi_{I_i}$ . The following diagram shows the relationship between  $I_1, J_1, \dots, J_k$  and the isogenies corresponding to them.



Therefore, our task is to compute the codomain  $E_i$  of  $\varphi_{J_i}$ , the ideal  $I_i$ , and the images of a basis of  $E_0[m_1]$  under  $\varphi_{I_i}$  for  $i > 1$ . However, we require the images of a basis of  $E_0[m_1m_2]$  under  $\varphi_{I_i}$  for computing the next isogeny  $\varphi_{I_{i+1}}$  by EvalByKani. More precisely, our algorithm takes  $E_i, I_i, J_i, \varphi_{I_i}(P_0)$ , and  $\varphi_{I_i}(Q_0)$  as input and computes  $E_{i+1}, I_{i+1}, \varphi_{I_{i+1}}(P_0)$ , and  $\varphi_{I_{i+1}}(Q_0)$  as output. The outline of our algorithm is as follows:

1. Compute  $E_{i+1}, \varphi_{J_i} \circ \varphi_{I_i}(P_0)$ , and  $\varphi_{J_i} \circ \varphi_{I_i}(Q_0)$  by using Vélu’s formulas with input  $E_i$  and a generator of  $\ker \varphi_{J_i} = \varphi_{I_i}(E_0[I_iJ_i])$ .
2. Find  $\beta \in I_iJ_i$  and  $\alpha \in R$  such that  $q_{I_iJ_i}(\beta)$  is prime to  $m_1m_2$  and  $n(\alpha) + q_{I_iJ_i}(\beta) = m_2$ . Let  $I_{i+1}$  be  $\chi_{I_iJ_i}(\beta)$ .
3. Compute  $\varphi_{I_{i+1}}(m_1P_0)$  and  $\varphi_{I_{i+1}}(m_1Q_0)$  by  $m_1\varphi_{I_{i+1}} = \frac{1}{n(I_i)}\varphi_{J_i} \circ \varphi_{I_i} \circ \hat{\beta}$ .
4. Compute the image of a basis of  $E_{i+1}[m_1m_2]$  under  $\hat{\varphi}_{I_{i+1}}$  by EvalByKani with input  $m_2, E_0, E_{i+1}E_0, \varphi_{I_{i+1}}(m_1P_0), \varphi_{I_{i+1}}(m_1Q_0), \alpha(m_1P_0), \alpha(m_1Q_0)$ .
5. Compute  $\varphi_{I_{i+1}}(P_0)$  and  $\varphi_{I_{i+1}}(Q_0)$  by solving a discrete logarithm problem in  $E_0[m_1m_2]$ .

The following diagram shows the relationship between the ideals and the isogenies in the above algorithm. Here the dotted arrows represent the direction of the endomorphism  $\beta$  of  $E_0$ .



We give supplementary explanations for the above steps except for Step 1.

**Step 2.** As discussed in [27, §3.1], we can find elements in  $I_i J_i$  whose normalized norms are approximately  $\sqrt{p}$  by using lattice enumeration (e.g., see [7, Algorithm 2.7.5]) for  $I_i J_i$ . However, there exist exceptional cases that we will discuss later. Once we find many elements in  $I_i J_i$  whose normalized norms are approximately  $\sqrt{p}$ , we can find  $\beta$  such that  $m_2 - q_{I_i J_i}(\beta)$  is a prime number splitting in  $R$ . Then we can find  $\alpha$  by Cornacchia’s algorithm [7, Algorithm 1.5.2, 1.5.3]. This method is also used in SQIsignHD and SILBE for the case  $R = \mathbb{Z}[\sqrt{-1}]$ , i.e.,  $m_2 - q_{I_i J_i}(\beta)$  is the sum of two squares. However, these schemes do not use an endomorphism of  $E_0$ , instead they use an endomorphism of the abelian surface  $E_0^2$ .

**Step 3.** Since  $\beta = \hat{\varphi}_{I_{i+1}} \circ \varphi_{J_i} \circ \varphi_{I_i}$ , we have  $\varphi_{J_i} \circ \varphi_{I_i} \circ \hat{\beta} = m_1 n(I_i) \varphi_{I_{i+1}}$ . Therefore, we can compute  $\varphi_{I_{i+1}}(m_1 P_0)$  by  $\frac{1}{n(I_i)} \varphi_{J_i} \circ \varphi_{I_i} \circ \hat{\beta}(P_0)$  and  $\varphi_{I_{i+1}}(m_1 Q_0)$  similarly, where  $\frac{1}{n(I_i)}$  is the inverse of  $n(I_i)$  modulo  $m_2$ .

**Step 4.** We need to care about the fact that the output of EvalByKani could not be  $\hat{\varphi}_{I_{i+1}}$  but  $\iota_0 \circ \hat{\varphi}_{I_{i+1}}$  for some automorphism  $\iota_0$  of  $E_0$ . If the automorphism groups of  $E_0$  and  $E_{i+1}$  are  $\{\pm 1\}$ , then this does not cause any problem. This is because  $-\varphi_{I_{i+1}}$  is also an isogeny corresponding to  $I_{i+1}$ . However, if  $\iota_0$  is an isomorphism not in  $\{\pm 1\}$ , then the dual isogeny of  $\iota_0 \circ \hat{\varphi}_{I_{i+1}}$  does not correspond to  $I_{i+1}$ . This occurs when  $j(E_0) = 0$  or 1728. Therefore, we need to fix the post-composition by  $\iota_0$ . To do this, we evaluate  $\varphi_{J_i} \circ \varphi_{I_i}(P_0)$  in addition to a basis of  $E_{i+1}[m_1 m_2]$  by EvalByKani. By comparing the output with  $\beta(P_0)$ , we can determine  $\iota_0$ .

We also note that we want the codomain of the  $(m_2, m_2)$ -isogeny in EvalByKani in Step 4 to not be isomorphic to  $E_0 \times E_0$ . As discussed in [32, §2.4], this only occurs with a negligible probability for a cryptographic size of  $p$ . In addition, we can prove the following lemma.

**Lemma 1.** *We use the notation in the above setting. Suppose that  $E_0 \not\cong E_{i+1}$  and any  $\gamma \in \text{End}(E_0)$  whose norm is less than  $m_2$  is in  $R$ . Then the codomain of the  $(m_2, m_2)$ -isogeny in Step 4 is not isomorphic to  $E_0 \times E_0$ .*

*Proof.* Suppose that the codomain of the  $(m_2, m_2)$ -isogeny is isomorphic to  $E_0 \times E_0$ . Then we have the following commutative diagram:

$$\begin{array}{ccc} E_0 & \xrightarrow{\varphi_{I_{i+1}}} & E_{i_1} \\ \alpha \downarrow & & \downarrow \\ E_0 & \xrightarrow{\gamma} & E_0, \end{array}$$

where  $n(\gamma) = n(I_{i+1})$  and the right vertical arrow is an isogeny of degree  $n(\alpha)$ . Since  $\alpha$  commutes with  $\gamma$  and  $n(\alpha)$  is prime to  $n(I_{i+1})$  we have  $\varphi_{I_{i+1}} = \gamma$  up to post-composition of an isomorphism. This contradicts  $E_0 \not\cong E_{i+1}$ .  $\square$

The latter assumption in the lemma is satisfied if we take  $m_2 \approx \sqrt{p}$  because an element in  $\mathcal{O}_0 \setminus R$  has a norm approximately greater than  $p$ . If the first assumption is not satisfied, we can compute the image under  $\varphi_{I_{i+1}}$  directly since this is an endomorphism of  $E_0$ .

**Step 5.** Let  $(P_{i+1}, Q_{i+1})$  be a basis of  $E_{i+1}[m_1 m_2]$ . By Step 4, we know  $\hat{\varphi}_{I_i}(P_{i+1})$  and  $\hat{\varphi}_{I_i}(Q_{i+1})$ . Solving the discrete logarithm problems for these points with base  $P_0$  and  $Q_0$ , we have integers  $a, b, c, d$  such that

$$\hat{\varphi}_{I_i} \begin{pmatrix} P_{i+1} \\ Q_{i+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix}.$$

We evaluate the points on both sides of the above equation by  $\varphi_{I_i}$  and multiply them by the inverse of  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$  modulo  $m_1 m_2$ . Then we have

$$n(I_{i+1}) \begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} \begin{pmatrix} P_{i+1} \\ Q_{i+1} \end{pmatrix} = \begin{pmatrix} \varphi_{I_{i+1}}(P_0) \\ \varphi_{I_{i+1}}(Q_0) \end{pmatrix}.$$

### 3.3 Exceptional Cases

In this subsection, we discuss exceptional cases in Step 2 in the previous subsection. In particular, we consider what kind of ideals  $I_i J_i$  fail to find  $\beta$  and  $\alpha$ , and how to avoid them. To ease the notation, we denote  $I_i J_i$  by  $I$  and  $E_{i+1}$  by  $E$ .

**Exceptional Ideals.** Step 2 fails if the normalized norm of the smallest element in  $I$  is much smaller than  $\sqrt{p}$  and not prime to  $m_1 m_2$ . Consider this case and let  $(\beta_1, \beta_2, \beta_3, \beta_4)$  be the Minkowski-reduced basis of  $I$ . Then we have (see [27, §3.1])

$$p^2 \leq q_I(\beta_1)q_I(\beta_2)q_I(\beta_3)q_I(\beta_4) \leq 4p^2.$$

Since  $q_I(\beta_1) \ll \sqrt{p}$  and  $n(\mathbf{i})$  is small, we have  $\beta_2 = \gamma\beta_1$ , where  $\gamma$  is the smallest element in  $R \setminus \mathbb{Z}$ . Therefore, the elements in  $I$  whose normalized norms is smaller



than  $\sqrt{p}$  are of the form  $\delta\beta_1$  for  $\delta \in R$ . The normalized norms of these elements are divisible by  $q_I(\beta_1)$ , so not prime to  $m_1m_2$ .

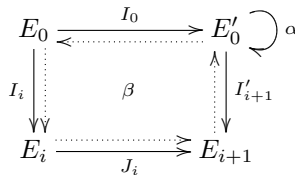
In terms of elliptic curves, the above exceptional case occurs when there exists an isogeny  $E_0 \rightarrow E$  whose degree is much smaller than  $\sqrt{p}$  and not prime to  $m_1m_2$ . Let  $n$  be a positive integer smaller than  $\sqrt{p}$ . Then the number of isogenies from  $E_0$  whose degree is smaller than  $n$  is approximately linear in  $n^2$ . Therefore, we can assume that the probability that  $I$  has an element whose normalized norm is smaller than  $n$  is approximately  $n^2/p$ . This probability is small but not negligible in practice. Especially, it occurs with a high probability in the case that  $J$  is a left  $\mathcal{O}_0$ -ideal, i.e.,  $I_1 = \mathcal{O}_0$  and  $E_1 = E_0$ . In this case, there exists the isogeny  $\varphi_{J_1} : E_0 \rightarrow E_2$  of degree  $m_1 < \sqrt{p}$ . Therefore, we need to avoid the exceptional cases.

**Avoiding Exceptional Cases.** To avoid the exceptional cases, we use other supersingular elliptic curves whose endomorphism rings contain an imaginary quadratic order with a small discriminant. A similar idea is used in the SQUISH. See §2.5.2 in the document in [6] for the details.

To explain our method, we define the following term.

**Definition 1 (connecting tuple).** For a positive integer  $N$ , an  $(E_0; P_0, Q_0)$ -connecting tuple is a tuple  $(E'_0, I_0, \mathfrak{D}, P'_0, Q'_0, \sigma(P'_0), \sigma(Q'_0))$ , where  $E'_0$  is a supersingular elliptic curve over  $\mathbb{F}_{p^2}$ ,  $I_0$  is a left  $\mathcal{O}_0$ -ideal such that  $\mathcal{O}_R(I_0) \cong \text{End}(E'_0)$  and  $\mathfrak{n}(I_0)$  is prime to the order of  $P_0$  and  $Q_0$ ,  $\mathfrak{D}$  is an imaginary quadratic order contained in  $\text{End}(E'_0)$ ,  $P'_0$  and  $Q'_0$  are the images of  $P_0$  and  $Q_0$  under  $\varphi_{I_0}$ , and  $\sigma$  is an element in  $\mathfrak{D}$  such that  $\mathfrak{D} = \mathbb{Z}[\sigma]$ .

Suppose that we are given an  $(E_0; P_0, Q_0)$ -connecting tuple  $(E'_0, I_0, \mathfrak{D}, P'_0, Q'_0, \sigma(P'_0), \sigma(Q'_0))$ . Instead of computing an isogeny between  $E_0$  and  $E_{i+1}$ , we compute an isogeny between  $E'_0$  and  $E_{i+1}$ . The following diagram shows the relationship between ideals and isogenies in this case.



In summary, we execute the following steps instead of Step 2–5 in the previous subsection:

- 2' Find  $\beta \in \bar{I}_0 I_i J_i$  and  $\alpha \in \mathfrak{D}$  such that  $q_{\bar{I}_0 I_i J_i}(\beta)$  is prime to  $m_1m_2$  and  $\mathfrak{n}(\alpha) + q_{\bar{I}_0 I_i J_i}(\beta) = m_2$ . Let  $I'_{i+1}$  be  $\chi_{\bar{I}_0 I_i J_i}(\beta)$ .
- 3' Compute  $\varphi_{I'_{i+1}}(m_1 P'_0)$  and  $\varphi_{I'_{i+1}}(m_1 Q'_0)$  by
 
$$m_1 \varphi_{I'_{i+1}} = \frac{1}{\mathfrak{n}(I_i)\mathfrak{n}(I_0)} \varphi_{J_i} \circ \varphi_{I_i} \circ \hat{\varphi}_{I_0} \circ \hat{\beta}.$$
- 4' Compute the image of a basis of  $E_{i+1}[m_1m_2]$  under  $\hat{\varphi}_{I'_{i+1}}$  by EvalByKani with input  $m_2, E'_0, E_{i+1}E'_0, \varphi_{I'_{i+1}}(m_1 P'_0), \varphi_{I'_{i+1}}(m_1 Q'_0), \alpha(m_1 P'_0), \alpha(m_1 Q'_0)$ .

5' Compute  $\varphi_{I'_{i+1}}(P'_0)$  and  $\varphi_{I'_{i+1}}(Q'_0)$  by solving a discrete logarithm problem in  $E'_0[m_1m_2]$ .

Consequently, we obtain a left  $\mathcal{O}_0$ -ideal  $I_0I'_{i+1}$  whose norm is prime to  $m_1m_2$  and the images of  $P_0$  and  $Q_0$  under  $\varphi_{I_0I'_{i+1}} = \varphi_{I'_{i+1}} \circ \varphi_{I_0}$ .

### 3.4 Explicit Algorithms

In this subsection, we give explicit ideal-to-isogeny algorithms. The first is an algorithm to compute an isogeny corresponding to an ideal of norm  $m_1$ , which is explained in the previous subsections. The second is an algorithm to compute an isogeny from an ideal by using the first algorithm repeatedly.

Let  $S_{ct}$  be a finite ordered set of  $(E_0; P_0, Q_0)$ -connecting tuples with different elliptic curves. We set the first entry of  $S_{ct}$  to be the trivial connecting tuple  $(E_0, \mathcal{O}_0, R, P_0, Q_0, \sigma(P_0), \sigma(Q_0))$ , where  $\sigma$  is a generator of  $R$ . We assume that  $S_{ct}$  is implicitly given as input for the algorithms in this section. A method to compute connecting tuples is given in Appendix A.

The first algorithm is given in Algorithm 2. We name this algorithm `ShortIdealTolsogenyIQO`. The second algorithm, `IdealTolsogenyIQO`, is given in Algorithm 3. This algorithm computes an isogeny corresponding to an ideal of norm  $m_1^k$ .

*Remark 1.* In our implementation, the imaginary quadratic orders in  $S_{st}$  are of the form  $\mathbb{Z}[\sqrt{-d}]$  for a small positive integer  $d$ . The elements in  $S_{st}$  are arranged in ascending order of  $d$ . Experimentally, the number of iterations of the for loop in line 4 in Algorithm 2 is one in most cases, and additional iterations are needed less than once in every ten runs. Over 10,000 runs of the algorithm for the characteristics in Sect. 5.1 showed that it is sufficient to have  $S_{st}$  of size five.

## 4 New Algorithm for SQIsign

As an application of our algorithms, we propose a new algorithm for SQIsign. Our algorithm for SQIsign uses `IdealTolsogenyIQO` instead of `IdealTolsogenyEichler`. This change replaces the computation of isogenies of higher degrees between elliptic curves with the computation of 2-isogenies and (2, 2)-isogenies and is expected to reduce the computational cost of SQIsign, but not affect its security and the size of public keys and signatures.

### 4.1 Setting

For the efficiency of our algorithm, we use powers of 2 as  $m_1$  and  $m_2$  and the theta algorithm by [11] for (2, 2)-isogenies. The theta algorithm requires points of order 8 instead of 2 to compute (2, 2)-isogenies. Therefore, we use  $p$  of the form  $2^{f_1+f_2+2}g - 1$  for small odd integer  $g$  and compute  $(2^{f_2}, 2^{f_2})$ -isogenies from points of order  $2^{f_2+2}$ .

In the following, we use elliptic curves over  $\mathbb{F}_{p^2}$  which are uniquely chosen from their isomorphic classes, e.g., “normalized Montgomery curves” obtained

---

**Algorithm 2: ShortIdealTolsogenyIQO**

---

**Input:** a supersingular elliptic curve  $E$ , a left  $\mathcal{O}_0$ -ideal  $I$  whose norm is prime to  $m_1m_2$ , a left  $\mathcal{O}_R(I)$ -ideal  $J$  of norm  $m_1$ , and  $\varphi_I(P_0), \varphi_I(Q_0)$ .

**Output:** The codomain of the isogeny  $\varphi_J, \beta \in IJ$  such that  $q_{IJ}(\beta)$  is prime to  $m_1m_2$ , and  $\varphi_{\chi_{IJ}(\beta)}(P_0), \varphi_{\chi_{IJ}(\beta)}(Q_0)$ .

```

1 Let  $K$  be a generator of  $\varphi_I(E_0[m] \cap E_0[IJ])$ ;
2 Compute an isogeny  $\varphi_J : E \rightarrow E'$  with kernel  $\langle K \rangle$ ;
3 Let  $S$  be  $\{P', Q'\}$ , where  $P', Q'$  a basis of  $E'[m_1m_2]$ ;
4 for  $(E'_0, I_0, \mathfrak{D}, P'_0, Q'_0, \sigma(P'_0), \sigma(Q'_0)) \in S_{\text{ct}}$  do
5   Search  $\alpha \in \mathfrak{D}$  and  $\beta \in \bar{I}_0IJ$  such that  $q_{\bar{I}_0IJ}(\beta)$  is prime to  $m_1m_2$  and
      $n(\alpha) + q_{\bar{I}_0IJ}(\beta) = m_2$ ;
6   if  $\alpha$  and  $\beta$  are found then
7     Let  $I' = \chi_{\bar{I}_0IJ}(\beta)$ ;
8     Let  $P_1 = \varphi_{I'}(m_1P'_0)$  and  $Q_1 = \varphi_{I'}(m_1Q'_0)$ ;
9     Let  $P_2 = \alpha(m_1P'_0)$  and  $Q_2 = \alpha(m_1Q'_0)$ ;
10    if  $j(E'_0) = 0$  or 1728 then
11      Append  $\varphi_J \circ \varphi_I \circ \hat{\varphi}_{I_0}(P'_0)$  to  $S$ ;
12      Let  $P'' = \beta(P'_0)$ ;
13    break;
14 Let  $S' = \text{EvalByKani}(m_2, E'_0, E', E'_0, P_1, Q_1, P_2, Q_2, S)$ ;
15 if  $\#S = 3$  then
16   Let  $P'''$  be the third element of  $S'$ ;
17   Compute  $\iota_0 \in \text{Aut}(E'_0)$  such that  $\iota_0(P''') = P''$ ;
18   Let  $S' = \iota_0(S')$ ;
19 Let  $P''_0, Q''_0$  be the first and second elements of  $S'$ ;
20 Find  $a, b, c, d \in \mathbb{Z}$  such that  $P''_0 = aP'_0 + bQ'_0$  and  $Q''_0 = cP'_0 + dQ'_0$ ;
21 Let  $M$  be the inverse of  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$  modulo  $m_1m_2$ ;
22 Let  $\begin{pmatrix} P \\ Q \end{pmatrix} = n(I')M \begin{pmatrix} P' \\ Q' \end{pmatrix}$ ;
23 return  $E', \beta, P, Q$ ;
```

---

by [6, Algorithm 1]. For such a curve  $E$ , we fix a basis of  $E[2^{f_1+f_2+2}]$  and denote it by  $(P_E, Q_E)$ .

Let  $E_0$  be a supersingular elliptic curve over  $\mathbb{F}_{p^2}$  of  $j$ -invariant 1728 and  $\iota : \mathcal{O}_0 := \left\langle 1, \mathbf{i}, \frac{1+\mathbf{j}}{2}, \frac{1+\mathbf{k}}{2} \right\rangle_{\mathbb{Z}} \rightarrow \text{End}(E_0)$  be an isomorphism. Let  $\lambda$  be a security parameter. For  $\lambda$ -bit security, our system parameters are  $p \approx 2^{2\lambda}$  of the above form,  $E_0, \iota$ , and a finite ordered set  $S_{\text{ct}}$  of  $(E_0; P_{E_0}, Q_{E_0})$ -connecting tuples with the first entry being the trivial connecting tuple  $(E_0, \mathcal{O}_0, \mathbb{Z}[\mathbf{i}], P_{E_0}, Q_{E_0}, \mathbf{i}(P_{E_0}), \mathbf{i}(Q_{E_0}))$ .

---

**Algorithm 3: IdealTolsogenyIQO**

---

**Input:** a supersingular elliptic curve  $E$ , a left  $\mathcal{O}_0$ -ideal  $I$  whose norm is prime to  $m_1 m_2$ , a left  $\mathcal{O}_R(I)$ -ideal  $J$  of norm  $m_1^k$ , and  $\varphi_I(P_0), \varphi_I(Q_0)$ .

**Output:** The codomain of the isogeny  $\varphi_J$ , a left  $\mathcal{O}_0$ -ideal  $I'$  such that  $I' \sim IJ$  and  $n(I')$  is prime to  $m_1 m_2$ , and  $\varphi_{I'}(P_0), \varphi_{I'}(Q_0)$ .

- 1 Let  $P = \varphi_I(P_0), Q = \varphi_I(Q_0)$ ;
  - 2 Let  $J_1, \dots, J_k$  be ideals such that  $n(J_i) = m_1$  and  $J = J_1 \cdots J_k$ ;
  - 3 **for**  $i = 1, \dots, k$  **do**
  - 4     Let  $E, \beta, P, Q = \text{ShortIdealTolsogenyIQO}(E, I, J_i, P, Q)$ ;
  - 5     Let  $I = \chi_{IJ_i}(\beta)$ ;
  - 6     Let  $J_j = \beta J_j \beta^{-1}$  for  $j = i + 1, \dots, k$ ;
  - 7 **return**  $E, I, P, Q$ ;
- 

### 4.2 Key Generation

Our key generation algorithm is almost the same as the SQISIGN. In particular, it is as follows:

- 1-3. The same as the key generation in Sect. 2.3.
4. Compute the codomain  $E_{\text{pub}}$  of  $\varphi_{I_{\text{sec}}}$  and the image of  $P_{E_0}$  and  $Q_{E_0}$  under  $\varphi_{I_{\text{sec}}}$  by IdealTolsogenyIQO.
5. Output a public key is  $E_{\text{pub}}$  and a secret key is  $(I_{\text{sec}}, \varphi_{I_{\text{sec}}}(P_{E_0}), \varphi_{I_{\text{sec}}}(Q_{E_0}))$ .

The product of the ideals in the input of the final call of ShortIdealTolsogenyIQO in IdealTolsogenyIQO in Step 4 above is equivalent to  $I_{\text{sec}}$ . Therefore, there exists an element  $\beta$  in this product whose normalized norm is prime to  $D_{\text{sec}}$  and we can compute  $\beta$  by the quaternions obtained by the calls of ShortIdealTolsogenyIQO other than the final call. Since  $D_{\text{sec}} < p^{1/4} \ll \sqrt{p} < 2^{f_2}$ , we can find an odd integer  $m$  and  $\alpha \in \mathbb{Z}[i]$  such that  $m^2 D_{\text{sec}} + n(\alpha) = 2^{f_2}$ . By using  $m\beta$  in the final call of ShortIdealTolsogenyIQO, we obtain the images of  $P_{E_0}$  and  $Q_{E_0}$  under  $m\varphi_{I_{\text{sec}}}$ . By dividing the images by  $m$  modulo  $2^{f_1+f_2+2}$ , we obtain the images of  $P_{E_0}$  and  $Q_{E_0}$  under  $\varphi_{I_{\text{sec}}}$ .

### 4.3 Commitment

To pull back a challenge isogeny to  $E_0$ , the degree of a commitment isogeny must be prime to the degree of the challenge isogeny. Since the degree of a challenge isogeny must be a power of 2 in our setting, the degree of a commitment isogeny must be odd. To achieve this, we use IdealTolsogenyIQO for commitments. Our commitment algorithm is as follows:

1. Sample a left  $\mathcal{O}_0$ -ideal  $J_{\text{com}}$  of norm  $2^{2\lambda}$  uniformly at random.
2. Compute the codomain  $E_{\text{com}}$  of  $\varphi_{J_{\text{com}}}$ , a left  $\mathcal{O}_0$ -ideal  $I_{\text{com}}$  such that  $I_{\text{com}} \sim J_{\text{com}}$  and  $n(I_{\text{com}})$  is odd, and the image of  $P_{E_0}$  and  $Q_{E_0}$  under  $\varphi_{I_{\text{com}}}$  by IdealTolsogenyIQO.
3. Output a commitment  $E_{\text{com}}$  and commitment key  $(I_{\text{com}}, \varphi_{I_{\text{com}}}(P_{E_0}), \varphi_{I_{\text{com}}}(Q_{E_0}))$ .

---

**Algorithm 4: Response**

---

**Input:** a public key  $E_{\text{pub}}$ , a secret key  $(I_{\text{sec}}, \varphi_{I_{\text{sec}}}(P_{E_0}), \varphi_{I_{\text{sec}}}(Q_{E_0}))$ , a commitment  $E_{\text{com}}$ , a commitment key  $(I_{\text{com}}, \varphi_{I_{\text{com}}}(P_{E_0}), \varphi_{I_{\text{com}}}(Q_{E_0}))$ , and a challenge  $c$ .

**Output:** A sequence of the kernels of isogenies of degree  $2^{2f_1}$ .

- 1 Compute the left  $\mathcal{O}_0$ -ideal  $I'_{\text{chall}}$  corresponding to an isogeny with kernel  $\langle \varphi_{I_{\text{com}}}(P_{E_{\text{com}}} + cQ_{E_{\text{com}}}) \rangle$ ;
  - 2 Let  $I_{\text{res}} = \bar{I}_{\text{sec}}(I_{\text{com}} \cap I'_{\text{chall}})$ ;
  - 3 Compute a left  $\mathcal{O}_R(I_{\text{sec}})$ -ideal  $J_{\text{res}}$  such that  $J_{\text{res}} \sim I_{\text{res}}$  and  $n(J_{\text{res}}) = 2^{2kf_1}$ ;
  - 4 Let  $J_1, \dots, J_k$  be ideals such that  $n(J_i) = 2^{f_1}$  and  $J_{\text{res}} = J_1 \cdots J_k$ ;
  - 5 Let  $E = E_{\text{pub}}$ ,  $I = I_{\text{sec}}$ ,  $P = \varphi_{I_{\text{sec}}}(P_{E_0})$ ,  $Q = \varphi_{I_{\text{sec}}}(Q_{E_0})$ ;
  - 6 Let  $S_{\text{res}} = \emptyset$ ;
  - 7 **for**  $i = 1, \dots, 2k - 2$  **do**
  - 8     **if**  $i$  *is odd* **then**
  - 9         Append a generator of  $\varphi_I(E_0[2^{2f_1}] \cap E_0[IJ_iJ_{i+1}])$  to  $S_{\text{res}}$ ;
  - 10     Let  $E, \beta, P, Q = \text{ShortIdealToIsogenyIQO}(E, I, J_i, P, Q)$ ;
  - 11     Let  $I = \chi_{IJ_i}(\beta)$ ;
  - 12     Let  $J_j = \beta J_j \beta^{-1}$  for  $j = i + 1, \dots, 2k$ ;
  - 13 Append a generator of  $\varphi_I(E_0[2^{2f_1}] \cap E_0[IJ_{2k-1}J_{2k}])$  to  $S_{\text{res}}$ ;
  - 14 **return**  $S_{\text{res}}$ ;
- 

#### 4.4 Challenge

Our challenge algorithm is almost the same as the SQISIGN, but we use only 2-isogenies. Consequently, a challenge is an integer  $c$  in  $[0, 2^\lambda)$ . This challenge corresponds to an isogeny with kernel  $\langle P_{E_{\text{com}}} + cQ_{E_{\text{com}}} \rangle$ .

#### 4.5 Response and Verification

In the SQISIGN, a response is computed by dividing into  $2^f$ -isogenies and represented by the sequence of their kernels. In our algorithm, a response is computed by dividing into  $2^{f_1}$ -isogenies and represented by the sequence of the kernels of the compositions of consecutive two  $2^{f_1}$ -isogenies. More precisely, for a chain of  $2^{f_1}$ -isogenies  $\varphi_k \circ \varphi_{k-1} \circ \dots \circ \varphi_1$ , our response is represented by the sequence of the kernels of  $\varphi_{2i} \circ \varphi_{2i-1}$  for  $i = 1, \dots, k/2$ . This is possible since  $E[2^{2f_1}]$  is  $\mathbb{F}_{p^2}$ -rational for elliptic curves  $E$  appearing in the response. This reduces the number of computations of torsion bases in verification and improves the efficiency of verification. In addition, we can compute the final two isogenies  $\varphi_k$  and  $\varphi_{k-1}$  without ShortIdealToIsogenyIQO since we do not need to compute the image of any point under the isogenies in the response.

Our response algorithm is given in Algorithm 4. Here we assume that the norm of the ideal obtained by the generalized KLPT algorithm is  $2^{2kf_1}$  for some integer  $k$ . Our verification algorithm is the same as the SQISIGN.

## 4.6 Other Methods

**Algorithm in QFESTA.** We can use `RandsogImages` by [32] instead of `IdealTolsogenyIQO` in the key generation and commitment algorithms. `RandsogImages` takes an integer  $d$  and points in  $E_0$  and outputs the codomain and the images of the points under a random  $d$ -isogeny. This algorithm is more efficient than `IdealTolsogenyIQO` because it requires only one  $(2^{f_1+f_2+2}, 2^{f_1+f_2+2})$ -isogeny while `IdealTolsogenyIQO` requires several  $(2^{f_2}, 2^{f_2})$ -isogenies (see Table 1 in the next section). However, the distribution of the codomain in the output of `RandsogImages` is not guaranteed to be uniform in the codomains of the  $d$ -isogenies from  $E_0$ .

In summary, this alternative method offers a trade-off between efficiency and security. This is the same situation as in an alternative method for the key generation in `SQISign` [13, Appendix D]. The security of this alternative method is discussed in [34]. We leave the detailed comparison between these methods in our algorithm for future work.

**Algorithm in DeuringVRF.** As mentioned in [29, §6], the ideal-to-isogeny algorithm in `DeuringVRF` could be applied to `SQISign`. Indeed, we can replace `ShortIdealTolsogenyIQO` with the ideal-to-isogeny algorithm in `DeuringVRF` (Algorithm 7 in [29]) in the `SQISign` algorithms explained in this section. In each call of the ideal-to-isogeny algorithm in `DeuringVRF`, we can compute an isogeny corresponding to an ideal of norm approximately  $p^{1/2}$ . This is almost the same as `IdealTolsogenyIQO`. On the other hand, the degree of the isogeny of dimension 2 in the ideal-to-isogeny algorithm in `DeuringVRF` is slightly larger than that in `ShortIdealTolsogenyIQO`. Therefore, simple replacement of `ShortIdealTolsogenyIQO` with the ideal-to-isogeny algorithm in `DeuringVRF` is not expected to improve the performance of `SQISign`.

## 5 Concrete Parameters and Efficiency

In this section, we propose concrete parameters for our algorithm for `SQISign` and discuss the efficiency of our algorithm compared with the `SQISIGN`.

### 5.1 Proposed Parameters

For the NIST security level 1, 3, and 5, we proposed the following parameters:

- For the security level 1:  $p = 2^{247} \cdot 79 - 1$ ,  $f_1 = 106$ ,  $f_2 = 139$ .
- For the security level 3:  $p = 2^{370} \cdot 231 - 1$ ,  $f_1 = 156$ ,  $f_2 = 212$ .
- For the security level 5:  $p = 2^{492} \cdot 539 - 1$ ,  $f_1 = 216$ ,  $f_2 = 274$ .

These primes have almost the same size as the primes in the `SQISIGN` corresponding to the same security levels.

**Table 1.** The numbers of  $T$ -isogenies and  $(2^f, 2^f)$ -isogenies in the key generation, commitment, and response in the SQISIGN and our algorithm.

	SQISIGN	Ours
	# of $T$ -isogenies	# of $(2^{f_2}, 2^{f_2})$ -isogenies
Key generation	16	7
Commitment	one $T/3^g$ -isogeny	3
Response	28	8

## 5.2 Efficiency

The main difference between our algorithm and the SQISIGN is that our algorithm uses `IdealTolsogenyIQO` instead of `IdealTolsogenyEichler`. Most of the computation time in `IdealTolsogenyIQO` is spent on the computation of  $(2^{f_2}, 2^{f_2})$ -isogenies. On the other hand, most of the computation time in `IdealTolsogenyEichler` is spent on the computation of  $T$ -isogenies. Therefore, we count the numbers of these isogenies in our algorithm and the SQISIGN.

In the following, we focus on the NIST security level 1 and claim that our algorithm is more efficient than the SQISIGN. At higher security levels, the advantage of our algorithm becomes more significant because the smoothness of  $T$  decreases as the security level increases.

At the current implementation of the SQISIGN for the NIST security level 1, the following parameters are used:  $f = 75$ , the norm of the output of the KLPT algorithm in the key generation is  $2^{675}$ , and  $n(I_{\text{rep}}) = 2^{1050}$ . Since `IdealTolsogenyEichler` requires two  $T$ -isogenies for each  $2^f$ -isogeny, the number of  $T$ -isogenies in the key generation is 16, and that in the response is 28. Note that the first  $2^f$ -isogeny in the key generation does not require any  $T$ -isogenies. In addition, the commitment algorithm requires one  $T/3^g$ -isogeny.

On the other hand, our algorithm uses  $2^{742}$  for the norm of the output of the KLPT algorithm in the key generation (we use  $742 = 7 \cdot f_1$  for simplicity) and the same norm for the response. Consequently, our algorithm requires 7  $(2^{f_2}, 2^{f_2})$ -isogenies in the key generation, 3 ( $= \lceil 256/f_1 \rceil$ )  $(2^{f_2}, 2^{f_2})$ -isogenies in the commitment, and 8 ( $= \lceil 1050/f_1 \rceil - 2$ )  $(2^{f_2}, 2^{f_2})$ -isogenies in the response.

Table 1 shows the numbers of isogenies in the key generation, commitment, and response in our algorithm and the SQISIGN.

We estimate the costs of a  $T$ -isogeny in the SQISIGN and a  $(2^{f_2}, 2^{f_2})$ -isogeny in our algorithm. Our estimation is based on the number of operations in  $\mathbb{F}_{p^2}$ . We use the number of multiplication (including squaring) in  $\mathbb{F}_p$  as the measure of the cost of an operation. We counted the number of multiplications by Python code.<sup>1</sup> In conclusion, we claim that the cost of a  $T$ -isogeny in the SQISIGN is at least 141,987  $\mathbb{F}_p$ -multiplications, and the cost of a  $(2^{f_2}, 2^{f_2})$ -isogeny in our algorithm is approximately 147,951  $\mathbb{F}_p$ -multiplications.

<sup>1</sup> The code is available at [https://github.com/hiroshi-onuki/SQISignIQO.jl/blob/main/measure\\_costs/measure\\_costs.py](https://github.com/hiroshi-onuki/SQISignIQO.jl/blob/main/measure_costs/measure_costs.py).

Together with the numbers of isogenies in Table 1, we conclude that our algorithm is at least twice as fast as the SQISIGN in the key generation and the signing.

Our algorithms is also more efficient in the verification than the SQISIGN because the numbers of separations in the response isogeny are reduced. As shown by Corte-Real Santos, Eriksen, Meyer, and Reijnders [37], reducing the number of separations in the response isogeny significantly improves the efficiency of the verification. In the NIST security level 1, the verification in the SQISIGN computes 14  $2^{75}$ -isogenies, while our algorithm computes 4  $2^{212}$ -isogenies and one  $2^{202}$ -isogeny. This reduce the number of the computations of deterministic torsion bases in the verification.

### 5.3 Implementation

We implemented our SQIsign by Julia language [4] with its computer algebra package Nemo [22]. Our code is available at

<https://github.com/hiroschi-onuki/SQIsignIQO.jl>.

Table 2 shows the computational times of key generation, signing, and verification in our implementation. The computational times are measured on a computer with an Intel Core i7-10700K CPU@3.70 GHz without Turbo Boost. The values are the averages of 100 runs.

For reference, we executed the benchmarking suite in the reference implementation of the SQISIGN on the same computer. Table 3 shows the results. The computational times are given in seconds by dividing the clock time by the CPU frequency.

**Table 2.** The computational times of key generation, signing, and verification in our implementation (sec.). The values are the averages of 100 runs.

	Key gen.	Sign	Verify
Level 1	1.88	3.41	0.24
Level 3	2.81	6.15	0.32
Level 5	4.73	8.84	0.50

**Table 3.** The computational times of key generation, signing, and verification in the reference implementation of the SQISIGN. The values are given in seconds by dividing the clock time by the CPU frequency and the average of 100 runs.

	Key gen.	Sign	Verify
Level 1	1.30	2.19	0.05
Level 3	9.34	17.20	0.32
Level 5	36.30	65.70	0.94



In the NIST security level 1, our implementation is not faster than the reference implementation of the SQISIGN. However, we believe that our algorithm outperforms the SQISIGN if we implement our algorithm in a lower-level language such as C or Rust. We leave such an implementation for future work.

In the NIST security level 3 and 5, our implementation is faster than the reference implementation of the SQISIGN. The advantage of our algorithm becomes more significant at higher security levels as we mentioned in Sect. 5.2.

## 6 Conclusion and Future Work

In this paper, we proposed a new ideal-to-isogeny algorithm using Kani’s reducibility theorem and embeddings of imaginary quadratic orders into the endomorphism rings of supersingular elliptic curves. Our algorithm works in the operations in  $\mathbb{F}_{p^2}$  if we use the characteristic  $p$  such that  $p + 1$  has a smooth divisor greater than  $\sqrt{p}$ . Especially, our algorithm is efficient if we use  $p$  of the form  $2^f g - 1$  for small odd integer  $g$ .

As an application of our algorithm, we proposed a new algorithm for SQISign. Our estimation shows that our algorithm is at least twice as fast as the SQISIGN in the key generation and the signing at the NIST security level 1. Our algorithm is also more efficient in the verification because the numbers of separations in the response isogeny are reduced. Notably, at higher security levels, the benefits of our algorithm become more pronounced. This assertion is substantiated by our implementation results.

Implementing our algorithm in a lower-level language such as C or Rust and comparing the efficiency with the SQISIGN are left for future work. Further improvements in the efficiency of our algorithm are also left for future work. For example, using a smooth factor of  $p - 1$  in the degree of the response Isogeny in addition to  $2^{f_1}$  may improve the efficiency of our algorithm since this could reduce the number of separations in the response isogeny.

**Acknowledgments.** We would like to thank Antonin Leroux for informing us of his ideal-to-isogeny algorithm. We also thank Andrea Basso, Luca De Feo, Pierrick Dartois, Antonin Leroux, Luciano Maino, and Benjamin Wesolowski for sharing their work on SQISign2D-West, and Max Duparc and Tako Boris Fouotsa for sharing their work on SQIPrime with us prior to publication. Finally, we thank the anonymous Asiacrypt 2024 reviewers for their valuable and constructive feedback.

## A Computing Connecting Tuples

Let  $\mathcal{O}_0, R, m_1, m_2, E_0, P_0, Q_0$  be as in Sect. 3. In this section, we explain how to compute  $(E_0; P_0, Q_0)$ -connecting tuples. In particular, we compute a tuple

$$(E'_0, I_0, \mathfrak{D}, P'_0, Q'_0, \sigma(P'_0), \sigma(Q'_0)),$$

where  $E'_0$  is a supersingular elliptic curve over  $\mathbb{F}_{p^2}$ ,  $I_0$  is a left  $\mathcal{O}_0$ -ideal such that  $\mathcal{O}_R(I_0) \cong \text{End}(E'_0)$  and  $\mathfrak{n}(I_0)$  is prime to  $m_1 m_2$ ,  $\mathfrak{D}$  is an imaginary quadratic order in  $\text{End}(E'_0)$ ,  $P'_0$  and  $Q'_0$  are the images of  $P_0$  and  $Q_0$  under  $\varphi_{I_0}$ , and  $\sigma$  is an element in  $\mathfrak{D}$  such that  $\mathfrak{D} = \mathbb{Z}[\sigma]$ .

### A.1 Algorithm for Computing Connecting Tuples

The outline of the computation of an  $(\mathcal{O}_0; P_0, Q_0)$ -connecting tuple is as follows:

1. Take a small square-free integer  $d$  such that  $p$  does not split in  $\mathbb{Q}(\sqrt{-d})$  and let  $\mathfrak{D}$  be  $\mathbb{Z}[\sqrt{-d}]$ .
2. Take a prime  $N > p$  such that there exists  $\sigma_0 \in \mathcal{O}_0$  such that  $\sigma_0^2 = -N^2d$ .
3. Let  $I_0 = \mathcal{O}_0\sigma_0 + \mathcal{O}_0N$ .
4. Compute the codomain  $E'_0$  of  $\varphi_{I_0}$ ,  $P'_0 = \varphi_{I_0}(P_0)$  and  $Q'_0 = \varphi_{I_0}(Q_0)$  by a variant of `IdealTolsogeny|QO`.
5. Compute  $\sigma(P'_0)$  and  $\sigma(Q'_0)$  by  $\sigma = \frac{1}{N^2}\varphi_{I_0} \circ \sigma_0 \circ \hat{\varphi}_{I_0}$ .

In the following, we explain the detail of each step.

**Step 1.** From the condition on  $d$ , there exists a supersingular elliptic curve over  $\mathbb{F}_{p^2}$  whose endomorphism ring containing a subring isomorphic to  $\mathbb{Z}[\sqrt{-d}]$  (see [28, Theorem 12 in Chap. 13]). The following steps computes such a curve  $E'_0$  and the connection between  $E_0$  and  $E'_0$ .

**Step 2.** Since  $N > p$ , there exists an  $N$ -isogeny from  $E_0$  to  $E'_0$  with high probability. Suppose such an isogeny  $\varphi$  exists. Let  $\sigma$  be an endomorphism on  $E'_0$  corresponding to  $\sqrt{-d}$ . Then  $\sigma_0 := \hat{\varphi} \circ \sigma \circ \varphi$  is an endomorphism on  $E_0$  corresponding to  $N\sqrt{-d}$ , i.e.,  $\sigma_0^2 = -N^2d$ .

A method to find such  $\sigma_0$  is as follows:

- (i) Let  $\bar{a}$  be a square root of  $N^2d/n(\mathbf{i})$  modulo  $p$  (such  $\bar{a}$  always exists from the condition on  $d$ ) and  $a$  be a lift of  $\bar{a}$  to  $\mathbb{Z}$  such that  $0 \leq a < p$ .
- (ii) Find  $\tau \in R$  such that  $n(\tau) = (N^2d - a^2n(\mathbf{i}))/p$  by Cornacchia's algorithm.
- (iii) Let  $\sigma_0 = a\mathbf{i} + \mathbf{j}\tau$ .

Since  $\text{tr}(\sigma_0) = 0$  and  $n(\sigma_0) = N^2d$ , we have  $\sigma_0^2 = -N^2d$ . The Step (ii) can fail. In this case, we replace  $N$  by another prime and retry.

**Step 3.** Let  $\sigma = \frac{1}{N}\sigma_0$ . Since  $I_0\sigma \subset I_0$ , we have  $\sigma \in \mathcal{O}_R(I_0)$ . Therefore, the endomorphism ring of the codomain of  $\varphi_{I_0}$  contains a subring isomorphic to  $\mathbb{Z}[\sqrt{-d}]$ . It holds that  $n(I_0) = N$ , which we will prove in the next subsection.

**Step 4.** As we mentioned in Sect. 3.3, the first `ShortIdealTolsogeny|QO` in the computation of  $\varphi_{I_0}$  may fail if we do not use connecting tuples.

To avoid this problem, we use the other factors of  $p^2 - 1$ . Let  $m_3$  be a smooth factor of  $p^2 - 1$  such that  $m_1m_3 > \sqrt{p}$  and  $m_3$  is prime to  $m_1m_2$ . Then we compute  $m_3$ -isogenies from  $E_0$  efficiency by using the method in [9]. In particular, we use  $J$  such that  $J \sim I_0$  and  $n(J) = m_3m_1^k$  and decompose  $J$  into  $J_1 \dots J_k$  such that  $n(J_1) = m_1m_3$  and  $n(J_i) = m_1$  for  $i = 2, \dots, k$ . Based on this decomposition, we compute the codomain of  $\varphi_J$  and the images of  $P_0$  and  $Q_0$  under  $\varphi_I$  for a left  $\mathcal{O}_0$ -ideal  $I$  such that  $I \sim J$  and  $n(I)$  is prime to  $m_1m_2$

by using IdealTolsogenylQO. Note that IdealTolsogenylQO may fail even in this case. In this case, we return to Step 2 and retry.

We can compute  $\varphi_{I_0}(P_0)$  and  $\varphi_{I_0}(Q_0)$  by using  $\alpha \in I_0$  such that  $\chi_{I_0}(\alpha) = I$ , where  $\alpha$  is obtained by the KLPT algorithm transforming  $I_0$  to  $J$  and the outputs of ShortIdealTolsogenylQO in the computation of  $\varphi_J$ . Since  $\alpha = \hat{\varphi}_I \circ \varphi_{I_0}$ , we have  $\varphi_{I_0} = \frac{1}{\deg \varphi_I} \varphi_I \circ \alpha$ .

**Step 5.** The isomorphism induced by  $\varphi_{I_0}$  maps  $\sigma$  to  $\frac{1}{N} \varphi_{I_0} \circ \frac{1}{N} \sigma_0 \circ \hat{\varphi}_{I_0}$ . Therefore, we have

$$\sigma(P'_0) = \sigma \circ \varphi_{I_0}(P_0) = \frac{1}{N} \varphi_{I_0} \circ \sigma_0(P_0).$$

The same holds for  $\sigma(Q'_0)$ .

### A.2 Proof of $n(I_0) = N$

We prove the following proposition, which we used in the above explanation.

**Proposition 1.** *The ideal  $I_0$  in the Step 3 in the previous subsection satisfies  $n(I_0) = N$ .*

First, we recall basic facts on quaternion algebras. We refer to [40, Chapter 15] for the details. For a fractional ideal  $I$  of  $\mathcal{B}_{p,\infty}$ , the *discriminant* of  $I$  is defined by

$$\text{disc}(I) = \det((\text{tr}(b_i b_j))_{i,j=1,\dots,4}),$$

where  $b_1, \dots, b_4$  is a  $\mathbb{Z}$ -basis of  $I$ . For fractional ideals  $I, J$  of  $\mathcal{B}_{p,\infty}$  such that  $I \subset J$ , we have  $\text{disc}(I) = [J : I]^2 \text{disc}(J)$ . Let  $\mathcal{O}$  be a maximal order of  $\mathcal{B}_{p,\infty}$  and  $I$  be a left  $\mathcal{O}$ -ideal. Then  $\text{disc}(\mathcal{O}) = p^2$  and  $\text{disc}(I) = n(I)^4 p^2$ .

Next, we prove the following lemmas used in the proof of Proposition 1.

**Lemma 2.** *Let  $\mathcal{O}_0$  be a special extremal order in  $\mathcal{B}_{p,\infty}$  and  $N > p$  be a prime. Let  $\alpha$  be an element in  $\mathcal{B}_{p,\infty}$  of the form  $\alpha = a + \mathbf{b}i + \mathbf{c}j + \mathbf{d}k$  for  $a, c, d \in \mathbb{Q}$  and  $b \in \frac{1}{N}\mathbb{Z} \setminus \mathbb{Z}$ . Then  $\alpha \notin \mathcal{O}_0$ .*

*Proof.* Since  $\mathcal{O}_0$  is a special extremal order,  $\mathcal{O}_0$  contains a sub-lattice  $L := \mathbb{Z} + \mathbb{Z}\mathbf{i} + \mathbb{Z}\mathbf{j} + \mathbb{Z}\mathbf{k}$ . An easy computation shows that  $\text{disc}(L) = (4n(\mathbf{i}))^2$ . Therefore, we have  $[\mathcal{O}_0 : L] = 4n(\mathbf{i})$ . Since  $n(\mathbf{i})$  is 1, 2, or the smallest prime that is a quadratic non-residue modulo  $p$ , we have  $n(\mathbf{i}) < p < N$ . Therefore,  $4n(\mathbf{i})\alpha \notin L$ . This means that  $\alpha \notin \mathcal{O}_0$ . □

**Lemma 3.** *Let  $\mathcal{O}_0$  be a special extremal order in  $\mathcal{B}_{p,\infty}$  and  $N$  be a prime. Let  $\alpha$  be an element in  $\mathcal{O}_0 \setminus N\mathcal{O}_0$  of norm divisible by  $N$ . Then the norm of a left  $\mathcal{O}_0$ -ideal  $I_0 = \mathcal{O}_0\alpha + \mathcal{O}_0N$  is  $N$ .*

*Proof.* By definition of the norm of an ideal, we have  $n(I_0) \mid N^2$ . Let  $\beta \in I_0$ . Then  $\beta = \alpha\beta_1 + N\beta_2$  for some  $\beta_1, \beta_2 \in \mathcal{O}_0$ . Therefore,  $n(\beta) \equiv n(\alpha)n(\beta_1) \equiv 0 \pmod{N}$ . This implies that  $N \mid n(\beta)$ .

Therefore,  $n(I_0) = N$  or  $N^2$ . The inclusion  $\mathcal{O}_0N \subset I_0$  and  $n(\mathcal{O}_0N) = N^2$  imply that if  $n(I_0) = N^2$ , then  $I_0 = \mathcal{O}_0N$ . This contradicts the assumption that  $\alpha \notin N\mathcal{O}_0$ . Therefore, we have  $n(I_0) = N$ . □

*Proof (Proof of Proposition 1).* Since  $\sigma_0 = ai + bj + ck + dij$  with  $0 \leq a < p$ , we have  $\sigma_0 \notin N\mathcal{O}_0$  by Lemma 2. Therefore, we have  $n(I_0) = N$  by Lemma 3.  $\square$

## References

1. Basso, A., Feo, L.D., Dartois, P., Leroux, A., Maino, L., Pope, G., Robert, D., Wesolowski, B.: SQIsign2D-West: The fast, the small, and the safer. Cryptology ePrint Archive, Paper 2024/760 (2024), <https://eprint.iacr.org/2024/760>, <https://eprint.iacr.org/2024/760>
2. Basso, A., Maino, L., Pope, G.: FESTA: Fast encryption from supersingular torsion attacks. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part VII. LNCS, vol. 14444, pp. 98–126. Springer, Singapore (Dec 2023). [https://doi.org/10.1007/978-981-99-8739-9\\_4](https://doi.org/10.1007/978-981-99-8739-9_4)
3. Bernstein, D.J., De Feo, L., Leroux, A., Smith, B.: Faster computation of isogenies of large prime degree. In: Galbraith, S. (ed.) ANTS-XIV - 14th Algorithmic Number Theory Symposium. Proceedings of the Fourteenth Algorithmic Number Theory Symposium (ANTS-XIV), vol. 4, pp. 39–55. Mathematical Sciences Publishers, Auckland, New Zealand (2020). <https://doi.org/10.2140/obs.2020.4.39>, <https://hal.inria.fr/hal-02514201>
4. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM Review **59**(1), 65–98 (2017). <https://doi.org/10.1137/141000671>, <https://epubs.siam.org/doi/10.1137/141000671>
5. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 423–447. Springer, Cham (Apr 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_15](https://doi.org/10.1007/978-3-031-30589-4_15)
6. Chavez-Saab, J., Santos, M.C., De Feo, L., Eriksen, J.K., Hess, B., Kohel, D., Leroux, A., Longa, P., Meyer, M., Panny, L., Patranabis, S., Petit, C., Rodríguez Henríquez, F., Schaeffler, S., Wesolowski, B.: SQIsign. Tech. rep., National Institute of Standards and Technology (2023), available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures>
7. Cohen, H.: A Course in Computational Algebraic Number Theory, Graduate Texts in Mathematics, vol. 138. Springer Berlin, Heidelberg (2010)
8. Cosset, R., Robert, D.: Computing  $(l, l)$ -isogenies in polynomial time on Jacobians of genus 2 curves. Mathematics of Computation **84**(294), 1953–1975 (2015)
9. Costello, C.: B-SIDH: Supersingular isogeny Diffie-Hellman using twisted torsion. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 440–463. Springer, Cham (Dec 2020). [https://doi.org/10.1007/978-3-030-64834-3\\_15](https://doi.org/10.1007/978-3-030-64834-3_15)
10. Dartois, P., Leroux, A., Robert, D., Wesolowski, B.: SQISignHD: New dimensions in cryptography. Cryptology ePrint Archive, Paper 2023/436 (2023), <https://eprint.iacr.org/2023/436>, <https://eprint.iacr.org/2023/436>
11. Dartois, P., Maino, L., Pope, G., Robert, D.: An algorithmic approach to  $(2, 2)$ -isogenies in the theta model and applications to isogeny-based cryptography. Cryptology ePrint Archive, Paper 2023/1747 (2023), <https://eprint.iacr.org/2023/1747>, <https://eprint.iacr.org/2023/1747>
12. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: Compact post-quantum signatures from quaternions and isogenies. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 64–93. Springer, Cham (Dec 2020). [https://doi.org/10.1007/978-3-030-64837-4\\_3](https://doi.org/10.1007/978-3-030-64837-4_3)

13. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: compact post-quantum signatures from quaternions and isogenies. *Cryptology ePrint Archive*, Report 2020/1240 (2020), <https://eprint.iacr.org/2020/1240>
14. De Feo, L., Leroux, A., Longa, P., Wesolowski, B.: New algorithms for the deuring correspondence - towards practical and secure SQISign signatures. In: Hazay, C., Stam, M. (eds.) *EUROCRYPT 2023, Part V*. LNCS, vol. 14008, pp. 659–690. Springer, Cham (Apr 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_23](https://doi.org/10.1007/978-3-031-30589-4_23)
15. Delfs, C., Galbraith, S.D.: Computing isogenies between supersingular elliptic curves over  $\mathbb{F}_p$ . *DCC* **78**(2), 425–440 (2016). <https://doi.org/10.1007/s10623-014-0010-1>
16. Deuring, M.: Die typen der multiplikatorenringe elliptischer funktionenkörper. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* **14**, 197–272 (1941)
17. Duparc, M., Fouotsa, T.B.: SQIPrime: A dimension 2 variant of SQISignHD with non-smooth challenge isogenies. *Cryptology ePrint Archive*, Paper 2024/773 (2024). <https://eprint.iacr.org/2024/773>
18. Duparc, M., Fouotsa, T.B., Vaudenay, S.: SILBE: an updatable public key encryption scheme from lollipop attacks. *Cryptology ePrint Archive*, Paper 2024/400 (2024), <https://eprint.iacr.org/2024/400>, <https://eprint.iacr.org/2024/400>
19. Eisentraeger, K., Hallgren, S., Leonardi, C., Morrison, T., Park, J.: Computing endomorphism rings of supersingular elliptic curves and connections to pathfinding in isogeny graphs. In: Galbraith, S. (ed.) *ANTS-XIV - 14th Algorithmic Number Theory Symposium*. Proceedings of the Fourteenth Algorithmic Number Theory Symposium (ANTS-XIV), vol. 4, pp. 215–232. Mathematical Sciences Publishers, Auckland, New Zealand (Jun 2020). <https://doi.org/10.2140/obs.2020.4.215>
20. Eisenträger, K., Hallgren, S., Lauter, K.E., Morrison, T., Petit, C.: Supersingular isogeny graphs and endomorphism rings: Reductions and solutions. In: Nielsen, J.B., Rijmen, V. (eds.) *EUROCRYPT 2018, Part III*. LNCS, vol. 10822, pp. 329–368. Springer, Cham (Apr / May 2018). [https://doi.org/10.1007/978-3-319-78372-7\\_11](https://doi.org/10.1007/978-3-319-78372-7_11)
21. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) *CRYPTO'86*. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). [https://doi.org/10.1007/3-540-47721-7\\_12](https://doi.org/10.1007/3-540-47721-7_12)
22. Fieker, C., Hart, W., Hofmann, T., Johansson, F.: Nemo/hecke: Computer algebra and number theory packages for the julia programming language. In: *Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation*. pp. 157–164. ISSAC '17, ACM, New York, NY, USA (2017). <https://doi.org/10.1145/3087604.3087611>, <http://doi.acm.org/10.1145/3087604.3087611>
23. Galbraith, S.D., Petit, C., Silva, J.: Identification protocols and signature schemes based on supersingular isogeny problems. In: Takagi, T., Peyrin, T. (eds.) *ASIACRYPT 2017, Part I*. LNCS, vol. 10624, pp. 3–33. Springer, Cham (Dec 2017). [https://doi.org/10.1007/978-3-319-70694-8\\_1](https://doi.org/10.1007/978-3-319-70694-8_1)
24. Howe, E.W., Leprévost, F., Poonen, B.: Large torsion subgroups of split Jacobians of curves of genus two or three. *Forum Mathematicum* **12**(3), 315–364 (2000)
25. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.Y. (ed.) *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*. pp. 19–34. Springer, Berlin, Heidelberg (Nov / Dec 2011). [https://doi.org/10.1007/978-3-642-25405-5\\_2](https://doi.org/10.1007/978-3-642-25405-5_2)

26. Kani, E.: The number of curves of genus two with elliptic differentials. *Journal für die reine und angewandte Mathematik* **485**, 93–122 (1997). <https://doi.org/10.1515/crll.1997.485.93>
27. Kohel, D., Lauter, K., Petit, C., Tignol, J.P.: On the quaternion  $\ell$ -isogeny path problem. *LMS Journal of Computation and Mathematics* **17**(A), 418–432 (2014). <https://doi.org/10.1112/S1461157014000151>, <https://hal.archives-ouvertes.fr/hal-01257092>
28. Lang, S.: *Elliptic Functions*. Graduate texts in mathematics, Springer, 2nd edn. (1987), <https://books.google.co.jp/books?id=IxRZAAAAYAAJ>
29. Leroux, A.: Verifiable random function from the Deuring correspondence and higher dimensional isogenies. *Cryptology ePrint Archive*, Paper 2023/1251 (2023), <https://eprint.iacr.org/2023/1251>, <https://eprint.iacr.org/2023/1251>
30. Lubicz, D., Robert, D.: Computing isogenies between abelian varieties. *Compositio Mathematica* **148**(5), 1483–1515 (2012). <https://doi.org/10.1112/S0010437X12000243>
31. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) *EUROCRYPT 2023*, Part V. LNCS, vol. 14008, pp. 448–471. Springer, Cham (Apr 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_16](https://doi.org/10.1007/978-3-031-30589-4_16)
32. Nakagawa, K., Onuki, H.: QFESTA: Efficient algorithms and parameters for FESTA using quaternion algebras. *Cryptology ePrint Archive*, Paper 2023/1468 (2023), <https://eprint.iacr.org/2023/1468>, <https://eprint.iacr.org/2023/1468>
33. Nakagawa, K., Onuki, H.: SQISign2D-East: A new signature scheme using 2-dimensional isogenies. *Cryptology ePrint Archive*, Paper 2024/771 (2024), <https://eprint.iacr.org/2024/771>, <https://eprint.iacr.org/2024/771>
34. Onuki, H.: On the key generation in SQISign. *Banach Center Publications* **126**, 89–104 (2023). <https://doi.org/10.4064/bc126-6>
35. Petit, C., Smith, S.: An improvement to the quaternion analogue of the  $l$ -isogeny path problem (2018), conference talk at MathCrypt 2018
36. Robert, D.: Breaking SIDH in polynomial time. In: Hazay, C., Stam, M. (eds.) *EUROCRYPT 2023*, Part V. LNCS, vol. 14008, pp. 472–503. Springer, Cham (Apr 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_17](https://doi.org/10.1007/978-3-031-30589-4_17)
37. Corte-Real Santos, M., Eriksen, J.K., Meyer, M., Reijnders, K.: AprèsSQI: Extra fast verification for SQISign using extension-field signing. In: Joye, M., Leander, G. (eds.) *EUROCRYPT 2024*, Part I. LNCS, vol. 14651, pp. 63–93. Springer, Cham (May 2024). [https://doi.org/10.1007/978-3-031-58716-0\\_3](https://doi.org/10.1007/978-3-031-58716-0_3)
38. Smith, B.A.: *Explicit endomorphisms and correspondences*. PhD thesis, University of Sydney (2005)
39. Vélú, J.: Isogénies entre courbes elliptiques. *Comptes-Rendus de l'Académie des Sciences* **273**, 238–241 (1971)
40. Voight, J.: *Quaternion Algebras*. Graduate Texts in Mathematics, Springer Cham (2022). <https://doi.org/10.1007/978-3-030-56694-4>



# SQIsign2D-East: A New Signature Scheme Using 2-Dimensional Isogenies

Kohei Nakagawa<sup>1</sup>(✉), Hiroshi Onuki<sup>2</sup>, Wouter Castryck<sup>3</sup>, Mingjie Chen<sup>3</sup>,  
Riccardo Invernizzi<sup>3</sup>, Gioella Lorenzon<sup>3</sup>, and Frederik Vercauteren<sup>3</sup>

<sup>1</sup> NTT Social Informatics Laboratories, Tokyo, Japan  
kohei.nakagawa@ntt.com

<sup>2</sup> The University of Tokyo, Tokyo, Japan

<sup>3</sup> COSIC, ESAT, KU Leuven, Leuven, Belgium

**Abstract.** Isogeny-based cryptography is cryptographic schemes whose security is based on the hardness of a mathematical problem called the isogeny problem, and is attracting attention as one of the candidates for post-quantum cryptography. A representative isogeny-based cryptography is the signature scheme called SQIsign, which was submitted to the NIST PQC standardization competition. SQIsign has attracted much attention because of its very short signature and key size among the candidates for the NIST PQC standardization. Recently, a lot of new schemes have been proposed that use high-dimensional isogenies. Among them, the signature scheme called SQIsignHD has an even shorter signature size than SQIsign. However, it requires 4-dimensional isogeny computations for the signature verification. In this paper, we propose a new signature scheme, SQIsign2D-East, which requires only two-dimensional isogeny computations for verification, thus reducing the computational cost of verification though increasing the signing cost. First, we generalized an algorithm called RandIsogImg, which computes a random isogeny of non-smooth degree. Then, by using this generalized RandIsogImg, we construct a new signature scheme SQIsign2D-East.

## 1 Introduction

In recent years, isogeny-based cryptography has been actively studied as one of the candidates for post-quantum cryptography (PQC). One of the representative isogeny-based cryptographies is the signature scheme called SQIsign [13], which was submitted to the NIST PQC standardization competition. SQIsign has attracted much attention because of its very short signature and key size among the candidates for the NIST PQC standardization. Another well-known isogeny-based cryptography is SIDH [20], which is proposed by De Feo and Jao. Additionally, SIKE [1], a key encapsulation scheme based on SIDH, remained

---

Initially, it was a paper by Nakagawa and Onuki, but the security issue described in Sect. 4.3 were pointed out by Wouter Castryck, Mingjie Chen, Riccardo Invernizzi, Gioella Lorenzon and Frederik Vercauteren and a solution was also proposed by them. Therefore, we merged these results into a single paper.

an alternative candidate for the NIST PQC standardization competition until Round 4. However, recent attacks [5, 24, 28] broke the security of SIDH and SIKE. These attacks find the secret isogeny from the two point images under the isogeny by computing high dimensional isogenies.

In response, a number of cryptographic applications of attacks on SIDH have been studied, such as SQIsignHD [11], FESTA [3], QFESTA [26] SCALLOP-HD [7], and IS-CUBE [25]. Among them, SQIsignHD is a variant of SQIsign that has a shorter signature size and higher signing performance than SQIsign. However, it requires 4-dimensional isogeny computations for signature verification, which leads to a large computational cost. Since NIST calls for signature schemes that have short signatures and fast verification, reducing the verification cost of SQIsignHD is an important issue.

## 1.1 Contributions

In this paper, we make the following contributions:

1. We construct a new algorithm `GenRandIsogImg`, which is a generalization of the algorithm called `RandIsogImg` proposed in [26], which computes the codomain and point images of a given degree isogeny from a *special* elliptic curve  $E_0$ . Our `GenRandIsogImg` computes the codomain and point images of a given degree isogeny from *any* elliptic curve  $E$ .
2. Using `GenRandIsogImg` as a building block, we propose a new variant of SQIsignHD, which only requires 2-dimensional isogeny computations for the verification. We name this signature scheme ‘SQIsign2D-East’.
3. We give concrete parameters of SQIsign2D for the NIST security level 1, 3, and 5. Under these parameter settings, we analyse the signature sizes and show that our signature sizes are smaller than SQIsign and larger than SQIsignHD.
4. We analyse the computational cost of SQIsign2D-East under the parameter for the NIST security level 1 and show that the verification cost of SQIsign2D is smaller than that of SQIsignHD though the signing cost is larger.

## 1.2 Related Works

At the same time as this work, [2] and [17] also proposed a variant of SQIsignHD based on 2-dimensional isogenies. The former is called ‘SQIsign2D-West’ and the later is called ‘SQIPrime’. These protocols are similar to ours, but they were proposed independently of us. Our protocol has a stronger security assumption than their protocol but seems to be more efficient. We leave the comparison with their protocol as future work.

Recently, [27] proposed an algorithm called `IdealToIsogenyIQ0` that makes the key generation and the signing procedure in SQIsign at least twice as fast. However, their costs are still larger than SQIsignHD and SQIsign2D-East as described in their paper.



### 1.3 Organization

In Sect. 2, we give some notation and background knowledge used in our protocol. In Sect. 3, we construct a generalized `RandIsogImg`. In Sect. 4, we propose our new signature scheme `SQIsign2D-East` and its security is analysed in Sect. 5. In Sect. 6, we give some concrete parameters for `SQIsign2D-East` and analyse the data size and the computational cost of `SQIsign2D-East`. Finally, in Sect. 7, we give the conclusion of this paper.

## 2 Preliminaries

In this section, we summarize some background knowledge used in our protocol.

### 2.1 Notation

Throughout this paper, we use the following notation. We let  $p$  be a prime number of cryptographic size, i.e.,  $p$  is at least about  $2^{256}$  and let  $\lambda$  be a security parameter. Let  $f(x)$  and  $g(x)$  be real functions. We write  $f(x) = O(g(x))$  if there exists a constant  $c \in \mathbb{R}$  such that  $f(x)$  is bounded by  $c \cdot g(x)$  for sufficiently large  $x$ . For a finite set  $S$ , we write  $x \in_U S$  if  $x$  is sampled uniformly at random from  $S$ . Let  $\perp$  be the symbol indicating failure of an algorithm.

### 2.2 Abelian Varieties and Isogenies

In this paper, we mainly use principally polarized superspecial abelian varieties of dimension one or two defined over a finite field of characteristic  $p$ . Such a variety is isomorphic to a supersingular elliptic curve, the product of two supersingular elliptic curves, or a Jacobian of a superspecial hyperelliptic curve of genus two, and always has a model defined over  $\mathbb{F}_{p^2}$ . Therefore, we only consider varieties defined over  $\mathbb{F}_{p^2}$ .

**Basic Facts.** An *isogeny* is a rational map between abelian varieties which is a surjective group homomorphism and has finite kernel. The *degree* of an isogeny  $\varphi$  is its degree as a rational map and is denoted by  $\deg \varphi$ . An isogeny  $\varphi$  is *separable* if  $\#\ker \varphi = \deg \varphi$ . A separable isogeny is uniquely determined by its kernel up to post-composition of an isomorphism. For an isogeny  $\varphi : A \rightarrow B$  between principally polarized abelian varieties, there exists a unique *dual isogeny*  $\hat{\varphi}$  such that  $\hat{\varphi} \circ \varphi$  is equal to the multiplication-by- $\deg \varphi$  map on  $A$ .

Let  $\varphi : A \rightarrow B$ ,  $\psi : A \rightarrow C$ , and  $\psi' : B \rightarrow D$  be isogenies such that  $\deg \varphi$  is coprime to  $\deg \psi$ . If  $\ker \psi' = \varphi(\ker \psi)$  holds, we say that  $\psi'$  is the push-forward of  $\psi$  by  $\varphi$  and denote it by  $\psi' = [\varphi]_* \psi$ . Under the same situation, we say that  $\psi$  is the pull-back of  $\psi'$  by  $\varphi$  and denote it by  $\psi = [\varphi]^* \psi'$ .

Let  $A$  and  $B$  be principally polarized abelian varieties. If there exists an isogeny between  $A$  and  $B$  then the dimensions of  $A$  and  $B$  are the same. If  $A$  is

superspecial then there exists an isogeny between  $A$  and  $B$  if and only if  $B$  is a superspecial abelian variety of the same dimension as  $A$ .

Let  $A$  be a principally polarized abelian variety and  $\ell$  a positive integer. An  $\ell$ -isotropic subgroup of  $A$  is a subgroup of the  $\ell$ -torsion subgroup  $A[\ell]$  of  $A$  on which the  $\ell$ -Weil pairing is trivial. An  $\ell$ -isotropic subgroup  $G$  is *maximal* if there is no other  $\ell$ -isotropic subgroup containing  $G$ . A separable isogeny whose kernel is a maximal  $\ell$ -isotropic subgroup is called an  $\ell$ -isogeny if the dimension of the domain is one or an  $(\ell, \ell)$ -isogeny if the dimension of the domain is two.

Let  $E$  be an elliptic curve defined over  $\mathbb{F}_{p^2}$ . Among the isomorphism class of  $E$ , we can chose a Montgomery curve as a canonical representative by using [6, Algorithm 1]. We call this curve the *normalized curve* of  $E$ . In this paper, we assume that all elliptic curves are normalized. Moreover, we can compute a canonical basis of the  $n$ -torsion subgroup  $E[n]$  defined over  $\mathbb{F}_{p^2}$  by using [6, Algorithm 3]. Especially when  $n = 2^k$  for a positive integer  $k$ , we can compute a canonical basis of  $E[n]$  by the algorithm proposed in [9, Section 5.1].

**Computing Isogenies.** Let  $A$  be a principally polarized abelian variety,  $\ell$  a positive integer, and  $G$  a maximal  $\ell$ -isotropic subgroup of  $A$ .

If the dimension of  $A$  is one then we can compute an  $\ell$ -isogeny  $\varphi$  with kernel  $G$  by Vélú’s formulas [32]. More precisely, given  $A, \ell, G$ , Vélú’s formulas give a method to compute the codomain of  $\varphi$  in  $O(\ell)$  operations on a field containing the points in  $G$ . In addition, for additional input  $P \in A$ , we can compute  $\varphi(P)$  in  $O(\ell)$  operations on a field containing the points in  $G$  and  $P$ . These computational costs are improved to  $\tilde{O}(\sqrt{\ell})$  by Bernstein, De Feo, Leroux, and Smith [4].

For an isogeny  $\varphi : A \rightarrow B$ , we say that information  $\mathcal{I}_\varphi$  is an *efficient representation* of  $\varphi$  when we can compute  $\varphi(P)$  in polynomial time from a given point  $P \in A$  and the information  $\mathcal{I}_\varphi$ . For example, the tuple  $(A, \ell, G)$  described above is an efficient representation of  $\ell$ -isogeny  $\varphi : A \rightarrow B$  when  $\ell$  is smooth.

If  $A$  is the Jacobian of a hyperelliptic curve of genus two and  $\ell = 2$  then we can compute  $(2, 2)$ -isogeny by formulas in Smith’s Ph.D thesis [30]. Formulas of  $(2, 2)$ -isogenies for the case  $A$  is the product of two elliptic curves is given by Howe, Leprévost, and Poonen [19]. In 2023, more efficient formulas of  $(2, 2)$ -isogenies is proposed by Dartois, Maino, Pope, and Robert [12]. In addition, an efficient formulas of  $(3, 3)$ -isogenies is proposed by Corte-Real Santos, Costello and Smith [29]. An algorithm to compute  $(\ell, \ell)$ -isogenies for a general  $\ell$  was given by [10] and later improved by [23]. The computational cost of this algorithm is  $O(\ell^2)$  operations on a field containing the points in  $G$ .

### 2.3 Quaternion Algebras and the Deuring Correspondence

**Quaternion Algebras.** A *quaternion algebra* over  $\mathbb{Q}$  is a division algebra defined by  $\mathbb{Q} + \mathbb{Q}\mathbf{i} + \mathbb{Q}\mathbf{j} + \mathbb{Q}\mathbf{k}$  and  $\mathbf{i}^2 = a, \mathbf{j}^2 = b, \mathbf{ij} = -\mathbf{ji} = \mathbf{k}$  for  $a, b \in \mathbb{Q}^*$ . We denote it by  $H(a, b)$ . We say  $H(a, b)$  is *ramified* at a place  $v$  of  $\mathbb{Q}$  if  $H(a, b) \otimes_{\mathbb{Q}} \mathbb{Q}_v$  is not isomorphic to the algebra of the  $2 \times 2$  matrices over  $\mathbb{Q}_v$ . There exists a quaternion algebra ramified exactly at  $p$  and  $\infty$ . Such an algebra is unique up to isomorphism. We denote it by  $\mathcal{B}_{p, \infty}$ .

Let  $\alpha = x + y\mathbf{i} + z\mathbf{j} + t\mathbf{k} \in H(a, b)$  with  $x, y, z, t \in \mathbb{Q}$ . The *conjugate* of  $\alpha$  is  $x - y\mathbf{i} - z\mathbf{j} - t\mathbf{k}$  and denoted by  $\bar{\alpha}$ . The *reduced norm* of  $\alpha$  is  $\alpha\bar{\alpha}$  and denoted by  $n(\alpha)$ .

An *order*  $\mathcal{O}$  of  $H(a, b)$  is a subring of  $H(a, b)$  that is also a  $\mathbb{Z}$ -lattice of rank 4. This means that  $\mathcal{O} = \mathbb{Z}\alpha_1 + \mathbb{Z}\alpha_2 + \mathbb{Z}\alpha_3 + \mathbb{Z}\alpha_4$  for a basis  $\{\alpha_1, \alpha_2, \alpha_3, \alpha_4\}$  of  $H(a, b)$ . We denote such an order by  $\mathbb{Z}\langle\alpha_1, \alpha_2, \alpha_3, \alpha_4\rangle$ . An order  $\mathcal{O}$  is said to be *maximal* if there is no larger order that contains  $\mathcal{O}$ .

For a maximal order  $\mathcal{O}$ , an (integral) *left  $\mathcal{O}$ -ideal*  $I$  is a  $\mathbb{Z}$ -lattice of rank 4 satisfying  $I \subset \mathcal{O}$  and  $\mathcal{O} \cdot I \subset I$ . An *right  $\mathcal{O}$ -ideal* is similarly defined. For an ideal  $I$ , we denote its conjugate by  $\bar{I} = \{\bar{\alpha} \mid \alpha \in I\}$ . We denote by  $n(I)$  the *reduced norm* of ideal  $I$ , defined as (the unique positive generator of) the  $\mathbb{Z}$ -module generated by the reduced norms of the elements of  $I$ . A left  $\mathcal{O}$ -ideal  $I$  of integer norm can be written as  $I = \mathcal{O}\alpha + \mathcal{O}n(I)$  for some  $\alpha \in I$ . We denote such  $I$  by  $I = \mathcal{O}\langle\alpha, n(I)\rangle$ . The *ideal equivalence* denoted by  $I \sim J$  means that there exists  $\beta \in \mathcal{B}_{p,\infty}^*$  such that  $I = J\beta$ .

**Deuring Correspondence.** Deuring [16] showed that the endomorphism ring of a supersingular elliptic curve over  $\mathbb{F}_{p^2}$  is isomorphic to a maximal order of  $\mathcal{B}_{p,\infty}$  and gave a correspondence (the *Deuring correspondence*) where a supersingular elliptic  $E$  curve over  $\mathbb{F}_{p^2}$  corresponding to a maximal order isomorphic to  $\text{End}(E)$ .

Suppose  $p \equiv 3 \pmod{4}$ . This is the setting we use in our protocol. Then we can take  $\mathcal{B}_{p,\infty} = H(-1, -p)$  and an elliptic curve over  $\mathbb{F}_{p^2}$  with  $j$ -invariant 1728 is supersingular. Let  $E_0$  be the elliptic curve over  $\mathbb{F}_{p^2}$  defined by  $y^2 = x^3 + x$ . Then  $j(E_0) = 1728$ , so  $E_0$  is supersingular. We define endomorphisms  $\iota : (x, y) \mapsto (-x, \sqrt{-1}y)$  and  $\pi : (x, y) \mapsto (x^p, y^p)$  of  $E_0$ , where  $\sqrt{-1}$  is a fixed square root of  $-1$  in  $\mathbb{F}_{p^2}$ . The endomorphism ring of  $E_0$  is isomorphic to  $\mathcal{O}_0 := \mathbb{Z}\langle\mathbf{1}, \mathbf{i}, \frac{\mathbf{1}+\mathbf{j}}{2}, \frac{\mathbf{1}+\mathbf{k}}{2}\rangle$ . This isomorphism is given by  $\iota \mapsto \mathbf{i}$  and  $\pi \mapsto \mathbf{j}$ . From now on, we identify  $\text{End}(E_0)$  with  $\mathcal{O}_0$  by this isomorphism.

Some isogeny-based protocols, e.g., SQISign [13], need to compute the image under an element in  $\mathcal{O}_0$  represented by the coefficients with respect to the basis  $(\mathbf{1}, \mathbf{i}, \frac{\mathbf{1}+\mathbf{j}}{2}, \frac{\mathbf{1}+\mathbf{k}}{2})$ . Let  $P \in E_0(\mathbb{F}_{p^2})$  and  $\alpha = x + y\mathbf{i} + z\frac{\mathbf{1}+\mathbf{j}}{2} + t\frac{\mathbf{1}+\mathbf{k}}{2}$  for  $x, y, z, t \in \mathbb{Z}$ . Given  $P$  and  $x, y, z, t$ , one can compute  $\alpha(P)$  in  $O(\log \max\{|x|, |y|, |z|, |t|\})$  operations on  $\mathbb{F}_{p^2}$  and  $O(\log p)$  operations on  $\mathbb{F}_{p^4}$ . The latter operations on  $\mathbb{F}_{p^4}$  is necessary only for the case when the order of  $P$  is even. We need to compute  $\alpha(P_0)$  and  $\alpha(Q_0)$  for a fixed basis  $P_0, Q_0$  of  $E_0[2^a]$  for some integer  $a$  in our protocol. In this case, by precomputing the images of  $P_0$  and  $Q_0$  under  $\mathbf{i}, \frac{\mathbf{1}+\mathbf{j}}{2}$ , and  $\frac{\mathbf{1}+\mathbf{k}}{2}$ , we can compute  $\alpha(P_0)$  and  $\alpha(Q_0)$  by scalar multiplications by  $x, y, z, t$  and additions.

The Deuring correspondence also gives a correspondence between isogenies and ideals. Let  $E_1$  be a supersingular elliptic curve over  $\mathbb{F}_{p^2}$  and let  $\mathcal{O}_1$  be a maximal order of  $\mathcal{B}_{p,\infty}$  such that  $\mathcal{O}_1 \cong \text{End}(E_1)$ . Let  $\phi : E_1 \rightarrow E_2$  be an  $N$ -isogeny, then the isogeny  $\phi$  can be associated to a left  $\mathcal{O}_1$ -ideal  $I_\phi$ . This ideal  $I_\phi$  is also a right  $\mathcal{O}_2$ -ideal for a maximal order  $\mathcal{O}_2$  satisfying  $\mathcal{O}_2 \cong \text{End}(E_2)$ . Such an ideal  $I_\phi$  is called a *connecting ideal* from  $\mathcal{O}_1$  to  $\mathcal{O}_2$ . Furthermore, it is

known that its norm  $n(I_\phi)$  is equal to the degree  $N$  of  $\phi$ . The order  $\mathfrak{O}$  denoted by  $\mathfrak{O} = \mathcal{O}_1 \cap \mathcal{O}_2$  is called the *Eichler order* and  $\mathfrak{O} = \mathbb{Z} + I_\phi$  holds. Moreover, two isogenies  $\phi, \psi : E_1 \rightarrow E_2$  that have the same domain and codomain correspond to equivalent ideals  $I_\phi \sim I_\psi$ .

Let  $I_\tau$  be a connecting ideal of norm  $d$  from  $\mathcal{O}_0 \cong \text{End}(E_0)$  to  $\mathcal{O}_1 \cong \text{End}(E_1)$  and let  $\tau : E_0 \rightarrow E_1$  be the corresponding isogeny. In our protocol, we need to compute the image under an endomorphism  $\alpha_1 \in \text{End}(E_1)$  represented as an element  $\alpha \in \mathcal{O}_0 \cap \mathcal{O}_1$ . Since  $\alpha \in \mathcal{O}_0$ , we can compute the image under the corresponding endomorphism  $\alpha_0 \in \text{End}(E_0)$  as described above. Then, if the order  $n$  of  $P \in E_1$  is coprime to  $d$ , we can compute  $\alpha_1(P)$  as follows:

$$\alpha_1(P) = \frac{1}{d} \tau \circ \alpha_0 \circ \hat{\tau}(P),$$

where  $\frac{1}{d}$  is an inverse of  $d$  modulo  $n$ .

**Algorithms Using Quaternion Algebras.** As in the above, we let  $\mathcal{O}_0$  be the maximal order of  $\mathcal{B}_{p,\infty}$  with basis  $(1, \mathbf{i}, \frac{1+\mathbf{j}}{2}, \frac{1+\mathbf{k}}{2})$ . Here, we introduce some existing algorithms using quaternion algebras necessary for the construction of our SQISign2D-East. These algorithms are used in SQISign (see the official document [6] for details).

- **FullRepresentInteger** $_{\mathcal{O}_0}(M)$ : Take an integer  $M > p$  as input, output  $\alpha \in \mathcal{O}_0$  such that  $n(\alpha) = M$ .
- **EichlerModConstraint** $(I, \gamma, \delta)$ : Take a left- $\mathcal{O}_0$  ideal  $I$  of prime norm  $N$  and  $\gamma, \delta \in \mathcal{O}_0$  as input, output  $(C_0 : D_0) \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$  such that  $\gamma(C_0\mathbf{j} + D_0\mathbf{k})\delta \in \mathbb{Z} + I$ .
- **StrongApproximation** $_M(N, C_0, D_0)$ : Take integers  $M, N, C_0$  and  $D_0$  as input, output  $\mu \in \mathcal{O}_0$  such that  $n(\mu) = M$  and  $\mu = m(C_0\mathbf{j} + D_0\mathbf{k}) + N\mu_1$ , where  $m \in \mathbb{Z}$  and  $\mu_1 \in \mathcal{O}_0$ .

### 2.4 Computing Isogenies of Dimension One from Dimension Two

In this subsection, we give an algorithm to compute isogenies of dimension one by using an isogeny of dimension two, which is an important sub-algorithm for our protocol. This algorithm comes from recent attacks to SIDH by [5, 24, 28]. We use the following theorem, which is based on Kani’s criterion [21].

**Theorem 1 ([24, Theorem 1]).** *Let  $N_1, N_2$ , and  $D$  be pairwise coprime integers such that  $D = N_1 + N_2$ , and let  $E_0, E_1, E_2$ , and  $E_3$  be elliptic curves connected by the following diagram of isogenies:*

$$\begin{array}{ccc} E_0 & \xrightarrow{\psi_2} & E_2 \\ \psi_1 \downarrow & \nearrow f & \downarrow \psi'_1 \\ E_1 & \xrightarrow{\psi'_2} & E_3 \end{array}$$

where  $\psi'_2 \circ \psi_1 = \psi'_1 \circ \psi_2$ ,  $f = \psi_2 \circ \hat{\psi}_1$ ,  $\deg(\psi_1) = \deg(\psi'_1) = N_1$ , and  $\deg(\psi_2) = \deg(\psi'_2) = N_2$ . Then, the isogeny

$$\Phi = \begin{pmatrix} \hat{\psi}_1 - \hat{\psi}_2 \\ \psi'_2 & \psi'_1 \end{pmatrix} : E_1 \times E_2 \rightarrow E_0 \times E_3 \tag{1}$$

is a  $(D, D)$ -isogeny with respect to the natural product polarizations on  $E_1 \times E_2$  and  $E_0 \times E_3$ , and has kernel  $\{([N_2]P, f(P)) \mid P \in E_1[D]\}$ .

Conversely, a  $(D, D)$ -isogeny with kernel  $\{([N_2]P, f(P)) \mid P \in E_1[D]\}$  is of the form  $\iota \circ \Phi$  with an isomorphism  $\iota$  from  $E_0 \times E_3$ . To construct algorithms to evaluate the isogenies in the matrix in Eq. (1), we need to restrict the possibility of  $\iota$ . In particular, we assume that the codomain  $E_3$  of  $\psi'_1$  and  $\psi'_2$  is not isomorphic to  $E_0$ . This assumption is plausible because there exist about  $p/12$  supersingular elliptic curves over  $\mathbb{F}_{p^2}$  up to isomorphism and  $\psi'_1$  seems to be a random isogeny unless we intend to have  $E_1 \cong E_3$ . Under this assumption, an isomorphism from  $E_0 \times E_3$  is represented by  $\begin{pmatrix} \iota_0 & 0 \\ 0 & \iota_3 \end{pmatrix}$  or  $\begin{pmatrix} 0 & \iota_3 \\ \iota_0 & 0 \end{pmatrix}$ , where  $\iota_0$  is an isomorphism from  $E_0$  and  $\iota_3$  is an isomorphism from  $E_3$ . Since we assume that  $E_0$  and  $E_3$  are normalized, we can determine the codomain of  $\Phi$  in only two ways:  $E_0 \times E_3$  or  $E_3 \times E_0$ .

Using Theorem 1 and assuming the above assumption, we construct an algorithm to evaluate the isogenies in the matrix in Equation (1) by computing a  $(D, D)$ -isogeny. We denote the algorithm by **KaniCod**.

Let  $N_1, N_2$  be integers coprime with each other and  $D = N_1 + N_2$ . Let  $E_1, E_2$  supersingular elliptic curves over  $\mathbb{F}_{p^2}$ ,  $(P_1, Q_1)$  a basis of  $E_1[D]$ ,  $(P_2, Q_2)$  a basis of  $E_2[D]$ ,  $S_1$  a finite subset of  $E_1$ , and  $S_2$  a finite subset of  $E_2$ . If there exist isogenies  $\psi_1 : E_0 \rightarrow E_1$  and  $\psi_2 : E_0 \rightarrow E_2$  such that  $\deg \psi_1 = N_1$ ,  $\deg \psi_2 = N_2$ ,  $P_2 = \psi_2 \circ \hat{\psi}_1(P_1)$ , and  $Q_2 = \psi_2 \circ \hat{\psi}_1(Q_1)$ , then **KaniCod** with input  $(N_1, N_2, E_1, E_2, P_1, Q_1, P_2, Q_2; S_1; S_2)$  returns the curve  $E_0$ , the image of  $S_1$  under  $\hat{\psi}_1$ , and the image of  $S_2$  under  $\hat{\psi}_2$ . If such isogenies do not exist then **KaniCod** returns  $\perp$ . The procedure for **KaniCod** is as follows:

1. Compute a  $(D, D)$ -isogeny  $\Phi$  with kernel  $\langle ([N_2]P_1, P_2), ([N_2]Q_1, Q_2) \rangle$ .
2. If the codomain of  $\Phi$  is not the product of elliptic curves then return  $\perp$ .
3. Otherwise let  $F_1 \times F_2$  be the codomain of  $\Phi$ .
4. Let  $P'_1$  and  $Q'_1$  be first components of  $\Phi((P_1, O_{E_2}))$  and  $\Phi((Q_1, O_{E_2}))$ .
5. Compute the  $D$ -Weil pairings  $e_D(P_1, Q_1)$  and  $e_D(P'_1, Q'_1)$ .
6. If  $e_D(P_1, Q_1)^{N_1} = e_D(P'_1, Q'_1)$  then return  $F_1$  and the first components of  $\Phi((R_1, O_{E_2}))$  and  $\Phi((O_{E_1}, R_2))$  for  $R_1 \in S_1$  and  $R_2 \in S_2$ .
7. If  $e_D(P_1, Q_1)^{N_2} = e_D(P'_1, Q'_1)$  then return  $F_2$  and the second components of  $\Phi((R_1, O_{E_2}))$  and  $\Phi((O_{E_1}, R_2))$  for  $R_1 \in S_1$  and  $R_2 \in S_2$ .
8. Otherwise, return  $\perp$ .

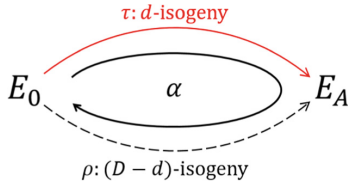
When  $D$  is smooth,  $P_1, Q_1 \in E_1(\mathbb{F}_{p^2})$ ,  $S_1 \subset E_1(\mathbb{F}_{p^2})$ ,  $P_2, Q_2 \in E_2(\mathbb{F}_{p^2})$ , and  $S_2 \subset E_2(\mathbb{F}_{p^2})$  the computational costs of **KaniCod** are  $O((\#S_1 + \#S_2) \log D)$  operations on  $\mathbb{F}_{p^2}$  by using the methods stated in Sect. 2.2. Especially,  $D$  is a power of 2 in our case.

### 2.5 RandIsogImg

Here, we recall the algorithm `RandIsogImg` which evaluates the codomain of a random isogeny of *non-smooth* degree and some point images under the isogeny. This algorithm was proposed in the paper of QFESTA [26] and is an important component of our SQIsign2D-East.

Let  $E_0$  be the elliptic curve over  $\mathbb{F}_{p^2}$  defined as  $E_0 : y^2 = x^3 + x$ . Let  $D$  be a smooth integer satisfying  $E_0[D] \subset E_0(\mathbb{F}_{p^2})$  and  $D \approx p$ , and let  $d$  be an integer coprime to  $D$  satisfying  $D - d \approx p$ . `RandIsogImg` takes integers  $d, D$  satisfying these conditions and a finite subset  $S$  of  $E_0$  as input, and outputs the codomain of a random  $d$ -isogeny  $\tau$  and the images of the points in  $S$  under  $\tau$ .

In this algorithm, we first compute an endomorphism  $\alpha \in \text{End}(E_0)$  of degree  $d \cdot (D - d)$  using `FullRepresentInteger` and decompose it into  $\alpha = \hat{\rho} \circ \tau$ , where  $\tau$  and  $\rho$  are the isogenies whose domains are  $E_0$  and whose degrees are  $d$  and  $D - d$ , respectively. (See the following diagram.) Since  $\deg \tau + \deg \rho = D$  and  $\gcd(\deg \tau, \deg \rho) = 1$ , we can evaluate point images under the isogeny  $\tau$  by using `KaniCod`. We describe the pseudo code of `RandIsogImg` in Algorithm 1.




---

#### Algorithm 1. `RandIsogImg` <sub>$\mathcal{O}_0$</sub> ( $d, D; S$ )

---

**Input:** Relatively prime integers  $d, D$  such that  $D - d \approx p$  and  $E_0[D] \subset E_0(\mathbb{F}_{p^2})$  and a finite subset  $S \subset E_0$ .

**Output:**  $(E_A, \tau(S))$  for a random  $d$ -isogeny  $\tau : E_0 \rightarrow E_A$ .

- 1: Let  $\alpha \leftarrow \text{FullRepresentInteger}_{\mathcal{O}_0}(d \cdot (D - d))$ .
  - 2: Take a basis  $P_0, Q_0$  of  $E_0[D]$ .
  - 3:  $(E_A, \tau(S), \emptyset) \leftarrow \text{KaniCod}(d, D - d, E_0, E_0, P_0, Q_0, \alpha(P_0), \alpha(Q_0); S; \emptyset)$ .
  - 4: **return**  $(E_A, \tau(S))$ .
- 

In addition, we can compute the left  $\mathcal{O}_0$ -ideal  $I_\tau = \mathcal{O}_0 \langle \alpha, d \rangle$ , which corresponds to the isogeny  $\tau$ . We denote the algorithm which outputs  $(E_A, \tau(S), I_\tau)$  by `RandIsogImgWithIdeal`.

## 2.6 SQIsignHD

SQIsignHD is a signature scheme proposed in [11] in 2023, which is based on SQIsign and utilizes an attack on SIDH to achieve a smaller signature length than SQIsign. There are two types of SQIsignHD, one using 4-dimensional isogenies and the other using 8-dimensional isogenies for the verification. In this section, we introduce an overview of SQIsignHD using 4-dimensional isogenies. For more details, we refer to [11].

First, we show the system parameters of SQIsignHD. Let  $a, b$  be integers satisfying  $2^a \approx 3^b \approx 2^\lambda$ , and let  $p$  be a prime satisfying  $p = 2^a 3^b f - 1$  for a sufficiently small integer  $f$ . Let  $E_0$  be the elliptic curve over  $\mathbb{F}_{p^2}$  defined as  $E_0 : y^2 = x^3 + x$ . Furthermore, we say that an odd integer  $q$  is  $2^a$ -good if there exist integers  $m_1, m_2$  satisfying  $m_1^2 + m_2^2 = 2^a - q$ .

SQIsignHD is obtained by applying the Fiat-Shamir transform [18] on the identification scheme based on the following diagram.

$$\begin{array}{ccc}
 E_0 & \xrightarrow[\text{com}]{\psi} & E_1 \\
 \downarrow \text{sk } \tau & & \downarrow \text{ch } \phi \\
 E_A & \xrightarrow[\text{resp}]{\sigma} & E_2
 \end{array}$$

In the following, we describe the overview of SQIsignHD identification protocol, which is similar to our protocol.

**keygen:** The prover generates a random  $3^{2b}$ -isogeny  $\tau : E_0 \rightarrow E_A$  and publishes the curve  $E_A$  as the public key.

**commit:** The prover generates a random  $3^{2b}$ -isogeny  $\psi : E_0 \rightarrow E_1$  and sends  $E_1$  to the verifier as the commitment.

**challenge:** The verifier generates a random  $3^b$ -isogeny  $\phi : E_1 \rightarrow E_2$  and sends it to the prover.

**response:** The prover computes the ideal  $J$  corresponding to  $\phi \circ \psi \circ \hat{\tau}$  and finds a random equivalent ideal  $I_\sigma \sim J$  whose norm  $q$  is  $2^a$ -good. Then, the prover sends to the verifier an efficient representation of the  $q$ -isogeny  $\sigma : E_A \rightarrow E_2$  corresponding to  $I_\sigma$ .

**verify:** The verifier checks that the response send by the prover correctly represents a  $q$ -isogeny  $\sigma : E_A \rightarrow E_2$ .

As an efficient representation of the  $q$ -isogeny  $\sigma$ , the prover sends  $(q, \sigma|_{E_A[2^a]})$ . Then, the verifier recovers the isogeny  $\sigma$  using Theorem 1. To apply Theorem 1, the verifier needs to compute a  $(2^a - q)$ -isogeny from  $E_A$ . However, this task is hard since the degree  $2^a - q$  is generally non-smooth. The verifier instead computes the 2-dimensional endomorphism over  $E_A \times E_A$  of degree  $2^a - q$  as follows:

1. Find two integers  $m_1, m_2$  satisfying  $m_1^2 + m_2^2 + q = 2^a$ .
2. Let  $\omega$  be the 2-dimensional endomorphism of degree  $m_1^2 + m_2^2 = 2^a - q$  defined as follows:

$$\omega = \begin{pmatrix} m_1 & -m_2 \\ m_2 & m_1 \end{pmatrix}.$$

Let  $I_2$  be the  $2 \times 2$  identity matrix. Under the following diagram, the verifier can recover  $\sigma$  by computing 4-dimensional  $2^a$ -isogeny. In this step, the verifier uses an extension of Theorem 1 to higher dimension by Robert [28].

$$\begin{array}{ccc} E_A \times E_A & \xrightarrow{\sigma I_2} & E_2 \times E_2 \\ \omega \downarrow & & \downarrow \omega' \\ E_A \times E_A & \xrightarrow{\sigma I_2} & E_2 \times E_2. \end{array}$$

### 3 Building Block for SQIsign2D-East

In this section, we give an algorithmic building block for SQIsign2D-East. We assume that we have a prime  $p = 2^{a+b}f - 1$  with  $a \approx b \approx \lambda$  and  $f \in \mathbb{N}$  as small as possible. We use the same notation  $q := \deg(\sigma)$  as in Subsect. 2.6. Note that the degree  $q$  is approximately  $p^{1/2}$ . In SQIsignHD, the verifier required a 4-dimensional isogeny computations since the auxiliary path  $\omega$  of degree  $2^a - q$  is a 2-dimensional isogeny. Our main idea is to generate the auxiliary path  $\omega$  as 1-dimensional isogeny of degree  $2^a - q$  by using `RandIsogImg`. However, the original `RandIsogImg` can only compute an isogeny from a specific elliptic curve  $E_0$ . Since the auxiliary path we need is the isogeny from the public key  $E_A$ , we have to construct a generalized `RandIsogImg`.

#### 3.1 Generalized `RandIsogImg`

We construct a generalized `RandIsogImg` so that we can compute an isogeny from arbitrary curves. Let  $E$  be an elliptic curve isogenous to  $E_0$  and let  $\mathcal{O} \cong \text{End}(E)$ . Let  $\tau$  be an  $N_\tau$ -isogeny from  $E_0$  to  $E$  and let  $I_\tau$  be a left  $\mathcal{O}_0$ -ideal corresponding to  $\tau$ . We propose an algorithm to compute an isogeny of non-smooth degree from  $E$ .

In the procedure of `RandIsogImg` $_{\mathcal{O}_0}(d, D; S)$ , we use  $\mathcal{O}_0$  only in step 1, where we find  $\alpha \in \mathcal{O}_0$  satisfying  $n(\alpha) = d \cdot (D - d)$ . Therefore, to construct a generalized `RandIsogImg`, we have to find  $\alpha \in \mathcal{O}$  satisfying  $n(\alpha) = d \cdot (D - d)$ . This can be achieved by using `EichlerModConstraint` and `StrongApproximation` as follows:

1. Using `EichlerModConstraint` $(I_\tau, 1, 1)$ , obtain  $(C_0 : D_0) \in \mathbb{P}^1(\mathbb{Z}/N_\tau\mathbb{Z})$  such that  $C_0j + D_0k \in \mathbb{Z} + I_\tau = \mathcal{O}_0 \cap \mathcal{O}$ .
2. Using `StrongApproximation` $_{d(D-d)}(N_\tau, C_0, D_0)$ , we can find  $\alpha \in \mathcal{O}_0 \cap \mathcal{O}$  satisfying  $n(\alpha) = d(D - d)$ .



The above approach is also used in the key generation and signing algorithm of SQISign [15]. Since we use **StrongApproximation**, the degree  $N_\tau$  of  $\tau$  must be prime and  $d(D - d) > pN_\tau^3$  must hold. If we assume that  $D - d \approx p$  as with the original **RandIsogImg**, the requirement on the degree  $d$  will be  $d > N_\tau^3$ . In addition, if we fix  $D$  around  $p$ , the condition  $D - d \approx p$  holds for almost all  $d$  satisfying  $d < D$ .

In the following, we show there is an additional hidden constraint on  $d$ . First, **StrongApproximation** $_{d(D-d)}(N_\tau, C_0, D_0)$  outputs  $\mu \in \mathcal{O}_0$  such that

$$n(\mu) = d(D - d) \text{ and } \mu = m(C_0\mathbf{j} + D_0\mathbf{k}) + N_\tau\mu_1,$$

where  $m \in \mathbb{Z}$  and  $\mu_1 \in \mathcal{O}_0$ . Therefore, the following equation holds:

$$d(D - d) = n(\mu) \equiv m^2p(C_0^2 + D_0^2) \pmod{N_\tau}.$$

For such an integer  $m$  to exist, the following condition must be satisfied:

$$\left(\frac{d(D - d)}{N_\tau}\right) = \left(\frac{p(C_0^2 + D_0^2)}{N_\tau}\right),$$

where  $(\cdot)$  is the quadratic residue symbol. On the other hand, from the definition of **EichlerModConstraint**, there exists an integer  $m'$  satisfying

$$m' + C_0\mathbf{j} + D_0\mathbf{k} \in I_\tau.$$

Hence, we have

$$n(m' + C_0\mathbf{j} + D_0\mathbf{k}) = (m')^2 + p(C_0^2 + D_0^2) \equiv 0 \pmod{N_\tau},$$

which means that

$$\left(\frac{p(C_0^2 + D_0^2)}{N_\tau}\right) = \left(\frac{-1}{N_\tau}\right).$$

Summarizing the above discussion,  $d$  must satisfy

$$\left(\frac{d(D - d)}{N_\tau}\right) = \left(\frac{-1}{N_\tau}\right). \tag{2}$$

However, if we use the degree  $d$  satisfying this condition in our protocol, we face a security issue. We explain this issue in Sect. 4.3. To avoid this security issue, we instead require that  $3 \mid d(D - d)$  and that 3 is not a square modulo  $N_\tau$ , i.e., we require  $N_\tau \equiv 5, 7 \pmod{12}$ . Then, our two new conditions together allow us to modify as follows:

- if  $d$  satisfies condition (2), then we call **StrongApproximation** with target norm  $M = d(D - d)$ ;
- otherwise, we call **StrongApproximation** with target norm  $d(D - d)/3$  to obtain an endomorphism  $\alpha'$ . In this case we have that  $d(D - d)/3$  satisfies

$$\left(\frac{d(D - d)/3}{N_\tau}\right) = \left(\frac{-1}{N_\tau}\right).$$

After that, we compute a random degree 3 isogeny  $\alpha'' : E \rightarrow E''$  using Vélú's formulas and we compose it with  $\alpha'$  to finally obtain an isogeny  $\alpha$  of degree  $d(D - d)$  from  $E$  to  $E''$ .

From the above argument, a generalized **RandIsogImg** for  $E$  is as shown in Algorithm 2.

---

**Algorithm 2.** **GenRandIsogImg** $_{\tau, I_\tau}(d, D; S)$

---

**Input:** An isogeny  $\tau : E_0 \rightarrow E$  of prime degree  $N_\tau$ , its corresponding ideal  $I_\tau$ , relatively prime integers  $d, D$  such that  $3 \mid d(D - d)$ ,  $N_\tau^3 < d < D \approx p$ , and  $E[D] \subseteq E(\mathbb{F}_{p^2})$ , and a finite set  $S \subseteq E$ ,  
**Output:**  $(F; \iota(S))$  for a random  $d$ -isogeny  $\iota : E \rightarrow F$ .  
1:  $(C_0 : D_0) \leftarrow \text{EichlerModConstraint}(I_\tau, 1, 1)$   
2: Let  $P, Q$  be a basis of  $E[D]$ .  
3: **if**  $d$  satisfies  $\left(\frac{d(D-d)}{N_\tau}\right) = \left(\frac{-1}{N_\tau}\right)$  **then**  
4:    $\alpha \leftarrow \text{StrongApproximation}_{d(D-d)}(N_\tau, C_0, D_0)$   
5:    $(F; \iota(S); \emptyset) \leftarrow \text{KaniCod}(d, D - d, E, E, P, Q, \alpha(P), \alpha(Q); S; \emptyset)$   
6: **else**  
7:    $\alpha' \leftarrow \text{StrongApproximation}_{d(D-d)/3}(N_\tau, C_0, D_0)$   
8:    $\alpha'' \leftarrow$  random 3-isogeny  $E \rightarrow E''$ , computed using Vlu's formulas.  
9:    $\alpha \leftarrow \alpha'' \circ \alpha'$   
10:    $(F; \iota(S); \emptyset) \leftarrow \text{KaniCod}(d, D - d, E, E'', P, Q, \alpha(P), \alpha(Q); S; \emptyset)$   
11: **end if**  
12: **return**  $(F; \iota(S))$

---

### 3.2 Computing Auxiliary Path

Unfortunately, the requirement  $d > N_\tau^3$  is too strong to compute an auxiliary path of degree  $r = 2^a - q \approx p^{1/2}$ . To allow the use of smaller degree, we take the following approach:

1. Let  $D_1$  be a smooth integer such that  $r(D_1 - r) > N_\tau^3$  and  $r(D_1 - r) < D$ .
2. Compute a  $r(D_1 - r)$ -isogeny using **GenRandIsogImg**.
3. By computing a  $(D_1, D_1)$ -isogeny, obtain a  $r$ -isogeny.

Then, the lower bound of  $r$  decreases from  $N_\tau^3$  to approximately  $N_\tau^3/D_1$ .

*Remark 1.* Strictly speaking, the lower bound of  $r$  is  $B = D_1/2 - \sqrt{(D_1/2)^2 - N_\tau^3} = (D_1/2) \cdot (1 - \sqrt{1 - 4N_\tau^3/D_1^2})$ . Especially when  $D_1^2 \gg N_\tau^3$ , we have  $B \approx N_\tau^3/D_1$ , where we used  $\sqrt{1 - \epsilon} \approx 1 - \epsilon/2$  for  $\epsilon \ll 1$ .

Algorithm to compute an auxiliary path is given in Algorithm 3. Especially in our protocol, we use  $D_1 = 2^a \approx p^{1/2}$  and  $D = 2^{a+b} \approx p$ . Since the degree  $r = 2^a - q$  of the auxiliary path we need is around  $p^{1/2}$ , we have  $r(D_1 - r) \approx p$  for almost all  $r < D_1$ . Hence, the condition  $r(D_1 - r) > N_\tau^3$  is satisfied when  $N_\tau < p^{1/3}$ .

---

**Algorithm 3.**  $\text{AuxiliaryPath}_{\tau, I_\tau}(r, D_1, D; S)$

---

**Input:** An isogeny  $\tau : E_0 \rightarrow E$  of prime degree  $N_\tau$ , its corresponding ideal  $I_\tau$ , integers  $r, D_1, D$  such that  $\gcd(r, D_1 D) = 1$ ,  $N_\tau^3 < r(D_1 - r) < D \approx p$ ,  $3 \mid d(D - d)$  for  $d = r(D_1 - r)$ , and  $E[D] \subset E(\mathbb{F}_{p^2})$ , and a finite set  $S \subset E$ .

**Output:**  $(F; \omega(S))$  for a random  $r$ -isogeny  $\omega : E \rightarrow F$ .

- 1: Let  $P, Q$  be a basis of  $E[D_1]$ .
  - 2:  $(F'; \iota(P), \iota(Q)) \leftarrow \text{GenRandIsogImg}_{\tau, I_\tau}(r(D_1 - r), D; P, Q)$ .
  - 3:  $(F; \omega(S); \emptyset) \leftarrow \text{KaniCod}(r, D_1 - r, E, F', P, Q, \iota(P), \iota(Q); S; \emptyset)$ .
  - 4: **return**  $(F; \omega(S))$ .
- 

In the following, let  $M(q) := q(2^a - q)(2^{a+b} - q(2^a - q))$ . From the above argument, the requirements on the degree  $q$  are as follows:

$$\begin{aligned} q &\text{ is odd integer smaller than } 2^a, \\ q(2^a - q) &< 2^{a+b}, \\ 3 \mid M(q). \end{aligned}$$

**Definition 1.** We say that a positive integer  $q$  is ‘ $(2^a, 2^b)$ -nice’ if  $q$  is an odd integer smaller than  $2^a$  and satisfying  $q(2^a - q) < 2^{a+b}$ . In addition, we say that a positive integer  $q$  is ‘ $(2^a, 2^b)$ <sub>3</sub>-nice’ if  $q$  is  $(2^a, 2^b)$ -nice and satisfies  $3 \mid M(q)$ .

*Remark 2.* The odd integer  $q < 2^a$  is always  $(2^a, 2^b)$ -nice when  $a - b \leq 2$  from the following inequality:

$$q \cdot (2^a - q) = 2^{2a-2} - (2^{a-1} - q)^2 < 2^{2a-2} \leq 2^{a+b}.$$

*Remark 3.* From the following facts, the probability that  $3 \mid M(q)$  is  $2/3$  or  $1$ .

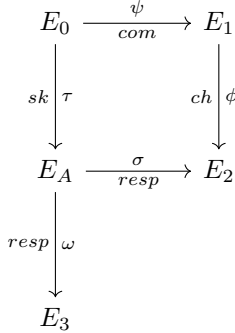
- if  $a \equiv b \pmod 2$  then  $3 \mid M(q)$ ,
- if  $a \equiv 0 \pmod 2$  and  $b \equiv 1 \pmod 2$  then  $3 \nmid M(q)$  if and only if  $q \equiv 2 \pmod 3$ ,
- if  $a \equiv 1 \pmod 2$  and  $b \equiv 0 \pmod 2$  then  $3 \nmid M(q)$  if and only if  $q \equiv 1 \pmod 3$ .

## 4 New Signature Scheme: SQIsign2D-East

In this section, we describe our new signature scheme SQIsign2D-East. First, we describe the detailed algorithm for SQIsign2D-East and then we propose its variant named ‘CompactSQIsign2D-East’, which has smaller signature size than the original SQIsign2D-East.

### 4.1 Description of SQIsign2D-East

We first describe the identification protocol underlying SQIsign2D-East. SQIsign-2D-East identification protocol is based on the following diagram.



We show the algorithms for the SQIsign2D-East identification scheme below.

**Parameter Setting.** The public parameter of SQIsign2D-East is taken as follows:

1. Let  $p$  be a prime of the form  $p = 2^{a+b}f - 1$ , where  $f$  is a small integer and  $a \approx b \approx \lambda$ .
2. Let  $E_0$  be the elliptic curve over  $\mathbb{F}_{p^2}$  defined as  $E_0 : y^2 = x^3 + x$ .
3. Let  $P_0, Q_0$  be a basis of  $E_0[2^{a+b}]$ .
4. Let  $\mathcal{O}_0 = \mathbb{Z}\langle 1, \mathbf{i}, \frac{1+\mathbf{j}}{2}, \frac{1+\mathbf{k}}{2} \rangle$ , which is isomorphic to  $\text{End}(E_0)$ .
5. Let  $\text{param} = (p, a, b, E_0, P_0, Q_0, \mathcal{O}_0)$ .

**Key Generation.** As we stated in Subsect. 3.2, we have to take the degree  $N_\tau$  of the secret isogeny  $\tau$  smaller than  $p^{1/3}$ . Fortunately, we can take  $N_\tau$  as small as approximately  $p^{1/4}$  while achieving  $\lambda$ -bits security as follows:

1. Take a random prime  $N_\tau < p^{1/4}$  such that  $\left(\frac{3}{N_\tau}\right) = -1$ .
2. Compute a random  $N_\tau$ -isogeny  $\tau : E_0 \rightarrow E$ .

The method to use a random degree smaller than  $p^{1/4}$  is also used in the key generation of SQIsign [13].

Since  $N_\tau$  is a large prime, we cannot compute  $\tau$  efficiently from  $\ker \tau$  using Vélu’s formulas. Instead, we compute an efficient representation  $(N_\tau, \tau(P_0), \tau(Q_0))$  of  $\tau$  using **RandIsogImg**. By using  $(N_\tau, \tau(P_0), \tau(Q_0))$ , we can efficiently compute  $\tau(T_0)$  for any  $T_0 \in E_0[2^{a+b}]$  as follows:

1. Find  $s, t \in \mathbb{Z}/2^{a+b}\mathbb{Z}$  such that  $T_0 = sP_0 + tQ_0$ .
2. Return  $\tau(T_0) = s\tau(P_0) + t\tau(Q_0)$ .

Now we show the key generation algorithm in Algorithm 4.

---

**Algorithm 4.**  $\text{keygen}(\text{param}) \rightarrow (pk, sk)$

---

**Input:** Public parameter  $\text{param} = (p, a, b, E_0, P_0, Q_0, \mathcal{O}_0)$ .

**Output:** Public key  $pk$  and secret key  $sk$ .

- 1: Take a random prime  $N_\tau < p^{1/4}$ .
  - 2:  $(E_A, R_A, S_A, I_\tau) \leftarrow \text{RandIsogImgWithIdeal}_{\mathcal{O}_0}(N_\tau, 2^{a+b}; P_0, Q_0)$ .
  - 3: **return**  $pk = E_A, sk = (\tau = (N_\tau, R_A, S_A), I_\tau)$ .
- 

**Commitment.** The commitment phase is similar to the key-generation. However, the commitment degree  $N_\psi$  need not to be prime smaller than  $p^{1/4}$  unlike  $N_\tau$ . Hence, we just chose a random odd integer  $N_\psi$  smaller than  $2^{a+b}$ .

As with the key generation, we compute  $(N_\psi, \psi(P_0), \psi(Q_0))$  as an efficient representation of  $\psi$  using  $\text{RandIsogImg}$ . As described above, we can efficiently evaluate  $\psi$  over the  $2^{a+b}$ -torsion subgroup using this representation. In addition, we can compute  $\hat{\psi}(T_1)$  for any  $T_1 \in E_1[2^{a+b}]$ , where  $E_1$  is the codomain of  $\psi$  as follows:

1. Find  $s, t \in \mathbb{Z}/2^{a+b}\mathbb{Z}$  such that  $T_1 = s\psi(P_0) + t\psi(Q_0)$ .
2. Return  $\hat{\psi}(T_A) = sN_\psi P_0 + tN_\psi Q_0$ .

Now, we show the commitment algorithm in Algorithm 5.

---

**Algorithm 5.**  $\text{commit}(\text{param}) \rightarrow (com, s)$

---

**Input:** Public parameter  $\text{param}$ .

**Output:** Commitment  $com$  and secret information  $s$ .

- 1: Take a random odd integer  $N_\psi < 2^{a+b}$ .
  - 2:  $(E_1, R_1, S_1, I_\psi) \leftarrow \text{RandIsogImgWithIdeal}_{\mathcal{O}_0}(N_\psi, 2^{a+b}; P_0, Q_0)$ .
  - 3: **return**  $com = E_1, s = (\psi = (N_\psi, R_1, S_1), I_\psi)$ .
- 

**Challenge.** We just compute a random  $2^b$ -isogeny from  $E_1$  using Vélu’s formulas. We show the challenge algorithm in Algorithm 6.

**Response.** In the response phase, we first compute the ideal  $I_\phi$ . This can be done by using  $\text{IsogToIdeal}$  algorithm [11, Algorithm 10], which takes two isogenies  $\psi : E_0 \rightarrow E_1$  and  $\phi : E_1 \rightarrow E_2$  and the ideal  $I_\psi$  corresponding to  $\psi$  as input and return the ideal  $I_\phi$  corresponding to  $\phi$ . Then, we compute the ideal  $J$

corresponding to  $\phi \circ \psi \circ \hat{\tau}$ . Next, we find all  $\alpha \in J$  such that  $q := n(\alpha)/n(J)$  is  $(2^a, 2^b)_3$ -nice by lattice enumeration (e.g., see [8, Algorithm 2.7.5]) and choose one of them uniformly. Then, we let  $I_\sigma = J \frac{\hat{\alpha}}{n(J)}$  and compute an efficient representation of the  $q$ -isogeny  $\sigma : E_A \rightarrow E_2$  corresponding to  $I_\sigma$ . Finally, we generate an auxiliary path  $\omega : E_A \rightarrow E_3$  and return an efficient representation of  $\sigma \circ \hat{\omega}$ .

---

**Algorithm 6.** challenge( $pk, \text{param}$ )  $\rightarrow ch$

---

**Input:** Public key  $pk$  and public parameter  $\text{param}$ .

**Output:** Challenge  $ch$ .

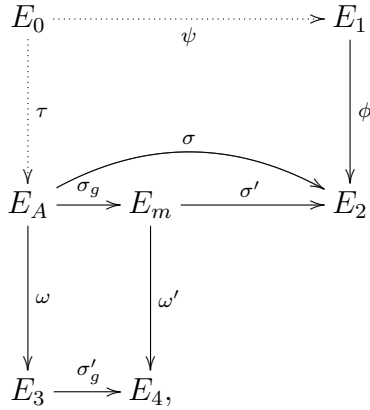
- 1: Take a random integer  $u \in_U \mathbb{Z}/2^b\mathbb{Z}$  and a bit  $\text{bin} \in_U \{0, 1\}$ .
  - 2: Let  $P'_1, Q'_1$  be the canonical basis of  $E_1[2^b]$ .
  - 3: If  $\text{bin} = 0$ ,  $K'_1 \leftarrow P'_1 + uQ'_1$ , otherwise,  $K'_1 \leftarrow uP'_1 + Q'_1$ .
  - 4: **return**  $ch = K'_1$ , a generator of the kernel of  $\phi : E_1 \rightarrow E_2$ .
- 

If there is no ideal  $I_\sigma$  whose norm  $q$  is  $(2^a, 2^b)_3$ -nice, we need to go back to the commitment phase. In the following, we discuss how to avoid this. From now on, we assume that  $a - b \leq 2$ , which means that at least  $2/3$  of odd integers smaller than  $2^a$  are  $(2^a, 2^b)_3$ -nice (see Remark 2 and Remark 3). To avoid the failure in finding  $I_\sigma$ , we consider using  $q' = q/\text{gcd}(q, f)$  instead of  $q$ . This reduces the constraint of  $q$  from  $q < 2^a$  to  $q' < 2^a \Leftrightarrow q < \text{gcd}(q, f) \cdot 2^a$ .

**Definition 2.** We say that a positive integer  $q$  is  $(2^a, 2^b, f)$ -nice when  $q' = q/\text{gcd}(q, f)$  is  $(2^a, 2^b)$ -nice. Similarly, we say that  $q$  is  $(2^a, 2^b, f)_3$ -nice when  $q' = q/\text{gcd}(q, f)$  is  $(2^a, 2^b)_3$ -nice.

Let  $\sigma$  be a  $q$ -isogeny computed in the response phase. Assume that  $q$  is  $(2^a, 2^b, f)_3$ -nice and let  $g = \text{gcd}(q, f)$ ,  $q' = q/g$ , and  $r = 2^a - q'$ . We formally decompose the  $q$ -isogeny  $\sigma$  to a  $g$ -isogeny  $\sigma_g : E_A \rightarrow E'_A$  and a  $q'$ -isogeny  $\sigma' : E'_A \rightarrow E_2$  and take the following procedures:

1. Compute  $\ker \sigma_g$  by evaluating  $\sigma$  over  $E_A[g]$ .
2. Compute  $\sigma_g : E_A \rightarrow E_m$  by using Vélú's formulas.
3. Obtain an  $r$ -isogeny  $\omega : E_A \rightarrow E_3$  by using AuxiliaryPath.
4. Let  $\sigma'_g = [\omega]_* \sigma_g$  and compute  $\ker \sigma'_g = \omega(\ker \sigma_g)$ .
5. Compute  $\sigma'_g : E_3 \rightarrow E_4$  by using Vélú's formulas.
6. Evaluate  $\sigma'$  and  $\omega'$  over  $E_m[2^a]$  by using the relationships:  $\sigma' \circ \sigma_g = \sigma$  and  $\omega' \circ \sigma_g = \sigma'_g \circ \omega$ .



Note that there is a concern that  $\deg \sigma_g = g$  is not coprime to  $\deg \omega = r$ . This means that the degree of  $\omega'$  may not be equal to  $r$  but reduces to  $\tilde{r} = r/h$  for a factor  $h$  of  $\gcd(g, r)$ . In this case, we additionally compute a random  $h$ -isogeny  $\iota$  from  $E_4$  and use  $\iota \circ \omega'$  as an auxiliary path. For simplicity, we consider the case  $h = 1$  in the following.

The response algorithm is given in Algorithm 7. To compute  $R'_2, S'_2$  in step 5, we use the following equation:

$$\sigma \circ [2^b] = \frac{1}{N_\tau N_\psi} \phi \circ \psi \circ \hat{\tau} \circ \hat{\alpha},$$

which is obtained by applying the Deuring correspondence on the equation  $I_\sigma = \bar{I}_\tau I_\psi I_\phi \cdot \frac{\hat{\alpha}}{N_\tau N_\psi 2^b}$ . Then, we can compute  $R'_2$  as follow:

$$R'_2 = \frac{1}{N_\psi N_\tau} \phi \circ \tau \circ \hat{\psi} \circ \hat{\alpha}(P_A) = \sigma(2^b P_A) = \sigma(P'_A).$$

We can compute  $S'_2$  similarly.

In step 6, we compute  $R'_3 = \omega(P'_A)$  and  $S'_3 = \omega(Q'_A)$  for an  $r$ -isogeny  $\omega : E_A \rightarrow E_3$ . Since  $K'_g = 2^a(R'_3 + \ell S'_3) = \omega(K_g)$  holds, we have  $\sigma'_g = [\omega]_* \sigma_g$ . Therefore, the following equation holds:

$$R'_4 = \sigma'_g(gR'_3) = \sigma'_g \circ \omega(gP'_A) = \omega' \circ \sigma_g(gP'_A) = \omega'(R'_m),$$

where  $\omega' = [\sigma_g]_* \omega$ . Similarly,  $S'_4 = \omega'(S'_m)$  also holds.

From the equation  $(P'_4, Q'_4) = (R'_4, S'_4)M = (\omega'(R'_m), \omega'(S'_m))M$  in step 11, the following equation holds:

$$(\hat{\omega}'(P'_4), \hat{\omega}'(Q'_4)) = r(R'_m, S'_m)M = -q(R'_m, S'_m)M,$$

---

**Algorithm 7.**  $\text{response}(sk, s, ch, \text{param}) \rightarrow \text{resp}$

---

**Input:** Secret key  $sk$ , secret information  $s$ , challenge  $ch$ , and public parameter  $\text{param}$ .

**Output:** Response  $\text{resp}$ .

- 1: Let  $I_\phi \leftarrow \text{IsogToIdeal}(\phi, \psi, I_\psi)$ .
  - 2: Let  $J = \bar{I}_\tau I_\psi I_\phi$ .
  - 3: Find all  $\alpha \in J$  such that  $q := n(\alpha)/n(J)$  is  $(2^a, 2^b, f)_3$ -nice by lattice enumeration and choose one of them uniformly.
  - 4: Let  $I_\sigma = J \frac{\bar{\alpha}}{n(J)}$ .
  - 5: Let  $q = n(I_\sigma)$ ,  $g = \gcd(q, f)$ ,  $q' = q/g$  and  $r = 2^a - q'$ .
  - 6: Let  $P_A, Q_A$  be the canonical basis of  $E_A[2^{a+b}g]$  and let  $(P'_A, Q'_A) = 2^b(P_A, Q_A)$ .
  - 7: Compute  $R'_2 = \sigma(P'_A)$  and  $S'_A = \sigma(Q'_A)$ .
  - 8: Let  $(E_3, R'_3, S'_3) \leftarrow \text{AuxiliaryPath}_{I_\tau}(r, 2^a, 2^{a+b}; P'_A, Q'_A)$ .
  - 9: Find an integer  $\ell$  such that  $2^a(R'_2 + \ell S'_2) = O$  (or  $2^a(\ell R'_2 + S'_2) = O$ ) and let  $K_g = 2^a(P'_A + \ell Q'_A)$  (or  $K_g = 2^a(\ell P'_A + Q'_A)$ ).
  - 10: Compute  $\sigma_g : E_A \rightarrow E_m = E_A/\langle K_g \rangle$ ,  $R'_m = \sigma_g(gP'_A)$ ,  $S'_m = \sigma_g(gQ'_A)$ .
  - 11: Let  $K'_g = 2^a(R'_3 + \ell S'_3)$  (or  $K'_g = 2^a(\ell P'_A + Q'_A)$ ).
  - 12: Compute  $\sigma'_g : E_3 \rightarrow E_4 = E_A/\langle K'_g \rangle$ ,  $R'_4 = \sigma'_g(gR'_3)$ ,  $S'_4 = \sigma'_g(gS'_3)$ .
  - 13: Let  $P'_4, Q'_4$  be the canonical basis of  $E_4[2^a]$  and compute the change of basis matrix  $M$  such that  $(P'_4, Q'_4) = (R'_4, S'_4)M$ .
  - 14: Compute  $(U_2, V_2) = -g(R'_2, S'_2)M$ .
  - 15: **return**  $\text{resp} = (K_g, E_4, U_2, V_2)$ .
- 

where we used  $r = 2^a - q' \equiv -q' \pmod{2^a}$ . By taking the image under the isogeny  $\sigma'$  of both sides, we obtain

$$\begin{aligned}
 (\sigma' \circ \hat{\omega}'(P'_4), \sigma' \circ \hat{\omega}'(Q'_4)) &= -q(\sigma'(R'_m), \sigma'(S'_m))M \\
 &= -q(\sigma' \circ \sigma_g(gP'_A), \sigma' \circ \sigma_g(gQ'_A))M \\
 &= -qg(\sigma(P'_A), \sigma(Q'_A))M \\
 &= -qg(R'_2, S'_2)M = q(U_2, V_2).
 \end{aligned}$$

Therefore, we obtain the following equation:

$$(U_2, V_2) = \left( \frac{1}{q} \sigma' \circ \hat{\omega}'(P'_4), \frac{1}{q} \sigma' \circ \hat{\omega}'(Q'_4) \right). \tag{3}$$

**Verify.** We show the verification algorithm in Algorithm 8. We prove that SQIsign2D-East identification protocol is complete. Assume here that the prover computes the response honestly. From Eq. 3, the subgroup  $K$  of  $E_A \times F$  satisfies



---

**Algorithm 8.**  $\text{verify}(pk, com, ch, resp, \text{param}) \rightarrow \text{accept/reject}$

---

**Input:** Public key  $pk$ , commitment  $com$ , challenge  $ch$ , response  $resp$ , and public parameter  $\text{param}$ .

**Output:** accept or reject.

- 1: Compute  $\sigma_g : E_A \rightarrow E_m = E_A/\langle K_g \rangle$ .
  - 2: Let  $P'_4, Q'_4$  be the canonical basis of  $E_3[2^a]$ .
  - 3: Compute a  $(2^a, 2^a)$ -isogeny  $\Phi : E_4 \times E_2 \rightarrow A$  with kernel  $K = \langle (P'_4, U_2), (Q'_4, V_2) \rangle$ .
  - 4: **if**  $A \cong E_m \times F$  for an elliptic curve  $F$  **then**
  - 5:   **return** accept.
  - 6: **else**
  - 7:   **return** reject.
  - 8: **end if**
- 

the following equation:

$$\begin{aligned} K &= \langle (P'_4, U_2), (Q'_4, V_2) \rangle \\ &= \left\langle \left( P'_4, \frac{1}{q} \sigma' \circ \hat{\omega}'(P'_4) \right), \left( Q'_4, \frac{1}{q} \sigma' \circ \hat{\omega}'(Q'_4) \right) \right\rangle \\ &= \langle (qP'_4, \sigma' \circ \hat{\omega}'(P'_4)), (qQ'_4, \sigma' \circ \hat{\omega}'(Q'_4)) \rangle. \end{aligned}$$

Let  $\sigma'' = [\omega']_* \sigma'$ ,  $\omega'' = [\sigma']_* \omega'$ , and  $F$  be the codomain of  $\sigma''$  and  $\omega''$ . From Theorem 1, a  $(2^a, 2^a)$ -isogeny  $\Phi$  with kernel  $K$  has the following form:

$$\Phi = \begin{pmatrix} \hat{\omega}' & -\hat{\sigma}' \\ \sigma'' & \omega'' \end{pmatrix} : E_4 \times E_2 \rightarrow E_m \times F$$

up to isomorphism. Therefore, the verifier accepts the honest response.

## 4.2 Reducing Signature Size

Applying the Fiat-Shamir transform, the signature of our protocol is made of the data  $(E_1, K_g, E_4, U_2, V_2)$ , where  $E_1$  is the commitment elliptic curve,  $E_4$  is the codomain of the auxiliary path,  $K_g \in E_A[g]$ , and  $U_2, V_2 \in E_2[2^a]$ .  $E_1$  and  $E_4$  can be determined by their  $j$ -invariant  $j(E_1), j(E_4) \in \mathbb{F}_{p^2}$ . Therefore, storing  $E_1$  and  $E_4$  takes  $2 \log_2 p^2 \approx 8\lambda$  bits. The points  $U_2$  and  $V_2$  can be compressed as in SIKE. Using this compression,  $U_2$  and  $V_2$  requires  $3a \approx 3\lambda$  bits. Similarly, the point  $K_g$  can be compressed and it requires about  $\log_2 g\lambda$  bits. Totally, the signature size is  $11\lambda$  bits.

Actually, we can reduce the signature size by about  $2\lambda$  bits by using the same method as SQIsign: include the information about  $\hat{\phi}$  instead of the commitment  $E_1$  in the signature. We name this variant 'CompactSQIsign2D-East'. To apply this method, we compute  $\omega'' = [\sigma']_* \omega'$  using `KaniCod`. Now we explain how

CompactSQIsign2D-East works. Let  $H : \{0, 1\}^* \times \mathbb{F}_{p^2} \rightarrow \mathbb{Z}/2^b\mathbb{Z} \times \{0, 1\}$  be a cryptographic hash function and let **GenKer** be an algorithm defined as follows:

**GenKer**( $m, E_1$ )  $\rightarrow K'_1$ :

1.  $h, \text{bin} \leftarrow H(m, j(E_1))$ .
2. Let  $P'_1, Q'_1$  be the canonical basis of  $E_1[2^b]$ .
3. If  $\text{bin} = 0$ , return  $K'_1 = hP'_1 + Q'_1$ .
4. Otherwise, return  $K'_1 = P'_1 + hQ'_1$ .

In the following, we regard  $\mathbb{F}_{p^2}$  as a totally ordered set under an appropriate order relation. We show the explicit algorithms for CompactSQIsign2D-East in Algorithm 9 and 10. Note that the key generation algorithm for CompactSQIsign2D-East is same as Algorithm 4.

---

**Algorithm 9. CompactSign**( $pk, sk, m, \text{param}$ )  $\rightarrow sig$

---

**Input:** The public key  $pk$ , the secret key  $sk$ , the message  $m$ , and the public parameter  $\text{param}$ .

**Output:** The signature  $sig$ .

- 1:  $(E_1, N_\psi, R_1, S_1, I_\psi) \leftarrow (\text{param})$ .
  - 2: Let  $K_1 \leftarrow \text{GenKer}(m, E_1)$  and  $\hat{\phi} : E_1 \rightarrow E_2$ .
  - 3: Let  $K_2$  be a generator of  $\ker \hat{\phi}$ .
  - 4: Find a  $2^b$ -torsion point  $P'_2$  linearly independent with  $K_2$  deterministically.
  - 5: Find  $t \in \mathbb{Z}/2^b\mathbb{Z}$  satisfying  $K_1 = t\hat{\phi}(P'_2)$ .
  - 6: Compute  $P'_4, Q'_4, R'_m, S'_m$ , and  $\text{resp} = (K_g, E_4, U_2, V_2)$  using Algorithm 7.
  - 7:  $(F; \emptyset; U, V) \leftarrow \text{KaniCod}(q', r, E_4, E_2, P'_4, Q'_4, qU_2, qV_2; \emptyset; R'_m, S'_m)$ .
  - 8: Let  $M$  and  $M_F$  be the Montgomery coefficient of  $E_m$  and  $F$ , respectively.
  - 9: **if**  $M \leq M_F$  **then**
  - 10:      $\text{bin} \leftarrow 0$ .
  - 11: **else**
  - 12:      $\text{bin} \leftarrow 1$ .
  - 13: **end if**
  - 14: **return**  $sig = (K_g, F, U, V, K_2, t, \text{bin})$ .
- 

Since the point  $K_2 \in E_2[2^b]$  can be represented by a single  $\mathbb{Z}/2^b\mathbb{Z}$  element, the size of  $(K_2, t)$  is about  $2b$  bits. Therefore, the total signature size is about  $\log_2 p^2 + 3a + 2b \approx 9\lambda$  bits.

### 4.3 Security Issue

We discuss the security issue when we use only  $(2^a, 2^b)$ -nice degrees  $q$  satisfying Eq. (2) for  $d = q(2^a - q)$  and  $D = 2^{a+b}$ .

As a first step, we observe that an adversary can evaluate  $\sigma$  at any input; the degree  $q$  can be then recovered using a pairing computation combined with

---

**Algorithm 10.** CompactVerify( $pk, m, sig, \text{param}$ )  $\rightarrow$  accept/reject

---

**Input:** The public key  $pk$ , the message  $m$ , the signature  $sig$ , and the public parameter  $\text{param}$ .

**Output:** accept or reject.

- 1: Let  $P_A, Q_A$  be the canonical basis of  $E_A[2^{a+b}g]$ .
  - 2: Compute  $\sigma_g : E_A \rightarrow E_m = E_A/\langle K_g \rangle$ ,  $R'_m = \sigma_g(2^b g P_A)$ ,  $S'_m = \sigma_g(2^b g Q_A)$ .
  - 3: Compute a  $(2^a, 2^a)$ -isogeny  $\Phi : E_m \times f \rightarrow A$  with kernel  $\langle (R'_m, U), (S'_m, V) \rangle$ .
  - 4: **if** not  $A \cong F_0 \times F_1$  for elliptic curves  $F_0$  and  $F_1$  **then**
  - 5:     **return** reject.
  - 6: **end if**
  - 7: Let  $M_0$  and  $M_1$  be the Montgomery coefficient of  $F_0$  and  $F_1$ , respectively.
  - 8: **if**  $M_0 > M_1$  **then**
  - 9:      $F_0, F_1 \leftarrow F_1, F_0$ .
  - 10: **end if**
  - 11:  $E_2 \leftarrow F_{\text{bin}_2}$ .
  - 12: Find a  $2^b$ -torsion point  $P'_2$  linearly independent with  $K_2$  deterministically.
  - 13: Compute a  $2^a$ -isogeny  $\hat{\phi} : E_2 \rightarrow E_1 = E_2/\langle K_2 \rangle$  and  $L_1 = \hat{\phi}(P'_2)$ .
  - 14: Let  $K_1 \leftarrow \text{GenKer}(m, E_1)$ .
  - 15: **if**  $K_1 = tL'_1$  **then**
  - 16:     **return** accept.
  - 17: **else**
  - 18:     **return** reject.
  - 19: **end if**
- 

an easy discrete log computation. Therefore it can be assumed that  $q$  is known. It is important to note that  $q$  varies with every signature and is essentially random subject to the above condition. Hence for each signature the adversary learns that  $M(q)$  has the same quadratic residuosity as  $-1 \pmod{N_\tau}$ . From Dirichlet's theorem on arithmetic progressions it follows that, as soon as  $M(q)$  is not an exact square, the density of primes  $N_\tau$  satisfying (2) is 50%. Thus, heuristically, we expect that  $N_\tau$  is uniquely determined by about  $\lambda/2$  values of  $q$ . This means that after seeing roughly  $\lambda/2$  signatures we should be able to find  $N_\tau$  by simply brute-forcing over all primes in  $(0, p^{1/4})$  and testing whether (2) holds for each of the corresponding values of  $q$ . Ignoring polynomial overhead, this step therefore has a complexity of  $O(2^{\lambda/2})$ .

Given the norm  $N_\tau$  of the secret ideal  $I_\tau$ , we can recover  $I_\tau$  by enumerating all left  $\mathcal{O}_0$ -ideals of norm  $N_\tau$  and check whether the corresponding isogenies have codomain isomorphic to  $E_A$ . There will be  $O(2^{\lambda/2})$  such ideals and they can be enumerated using the bijection from [22, Lemma 7.2]. Therefore, the cost of this step is  $\tilde{O}(2^{\lambda/2})$ .

### 4.4 On Sampling a Response Ideal

**Lemma 1.** *Let  $f \in \mathbb{Z}_{>0}$ . As  $x \rightarrow \infty$ , the proportion of integers  $q \in (x, fx)$  satisfying  $q < \gcd(q, f)x$  converges to*

$$\frac{P(f) - f}{f(f - 1)},$$

where  $P(f)$  denotes the gcd-sum function (also known as Pillai’s arithmetical function):

$$P(f) = \sum_{k=0}^{f-1} \gcd(k, f) = \sum_{d|f} d\varphi(f/d).$$

*Proof.* For any  $k = 0, \dots, f - 1$ , the number of integers  $q \in (x, fx)$  such that

$$q \bmod f = k \quad \text{and} \quad q < \gcd(q, f)x$$

is asymptotic to

$$\frac{1}{f} \cdot \frac{\gcd(k, f) - 1}{f - 1},$$

so the lemma follows by summing over all congruence classes mod  $f$ .

For a detailed study of the gcd-sum function, we refer to [31]. It is a multiplicative function which at prime powers  $f = \ell^e$  takes the values  $P(\ell^e) = (e + 1)\ell^e - e\ell^{e-1}$ . On “average”, it can be shown that

$$P(f) \approx \frac{3f \log f}{\pi^2},$$

although its concrete values fluctuate largely with  $f$ .

**Heuristic.** *Let  $J$  be a left ideal of  $\mathcal{O}_A$  and assume that  $0 \leq a - b \leq 2$ . If*

$$\delta\pi^2 2^{a-b} P(f) > f$$

where

$$\delta = \begin{cases} 1 & \text{if } a \equiv b \pmod{2}, \\ 2/3 & \text{if not} \end{cases}$$

(see Remark 3), then on average we expect there to exist at least one left ideal  $I_\sigma \sim J$  such that  $q := n(I_\sigma)$  is odd,  $M(q)$  is divisible by 3,  $q < \gcd(q, f)2^a$  and  $q(2^a - q) < 2^{a+b}$ . More quantitatively, the probability that no such ideal exists can be estimated as

$$\left(1 - \delta \frac{P(f)}{2f^2}\right)^{2\pi^2 f 2^{a-b}} \cdot \left(1 - \delta \frac{P(f) - f}{3f(f - 1)}\right)^{2\pi^2 f 2^{a-b}}.$$

*Explanation.* First note that the assumption  $0 \leq a-b \leq 2$  implies that  $q(2^a - q) < 2^{a+b}$  as soon as  $q$  is odd, so the last condition is of no concern. The Gaussian heuristic says that in any sufficiently general lattice  $\Lambda \subset \mathbb{R}^4$ , we expect

$$\#\{\alpha \in \Lambda \mid \|\alpha\| < R\} \approx \frac{\frac{\pi^2}{2} R^4}{\text{Vol}(\Lambda)},$$

where the numerator on the right is just the volume of a ball in  $\mathbb{R}^4$  with radius  $R$ . Applying this heuristic to  $\Lambda = J$ , which has Euclidean covolume  $n(J)^2 p/4$ , and to  $R = \sqrt{f 2^a n(J)}$ , we find an expected number of

$$\frac{2\pi^2 f^2 2^{2a}}{p} \approx 2\pi^2 f 2^{a-b}$$

elements  $\alpha \in J$  whose quaternion norm is smaller than  $f 2^a n(J)$ . Assuming  $\mathcal{O}_R(J)^\times = \{\pm 1\}$ , from [14, Lemma 1] it follows that there should be about  $\pi^2 f 2^{a-b}$  left ideals  $I_\sigma \subset \mathcal{O}_A$  satisfying  $I_\sigma \sim J$  and  $n(I_\sigma) < f 2^a$ .

If we furthermore assume that the norms of these  $I_\sigma$ 's behave as independent uniform variables in  $(0, f 2^a) \cap \mathbb{Z}$ , then we expect a proportion of  $1/2$  to be odd, a proportion of  $\delta$  to satisfy  $3 \mid M(q)$ , and a proportion of  $P(f)/f^2$  to meet the bound  $q < \text{gcd}(q, f) 2^a$ , leading to

$$\delta \frac{\pi^2 2^{a-b} P(f)}{f}$$

ideals whose norm  $q$  is of the desired type.

*Remark 4.* The count is slightly off in case  $f$  is even, because the condition  $q < \text{gcd}(q, f) 2^a$  is not independent of the condition that  $q$  is odd. A similar remark applies in case  $\delta = 2/3$  and  $3 \mid f$ . For simplicity, we ignore this issue here, although it is taken into account in the failure rates listed in Table 1.

A similar reasoning shows the failure rate of the signing procedure (i.e., the probability of having to go back to the commitment phase): this is

$$\left(1 - \delta \frac{P(f)}{2f^2}\right)^{2\pi^2 f 2^{a-b}}. \tag{4}$$

For the concrete parameter sets shown in Sect. 6.1, this gives (Fig. 1):

## 5 Security Analysis

We now discuss the security of our SQIsign2D-East.

NIST level	$a$	$b$	$f$	failure rate
1	127	126	27	$2^{-78}$
3	191	189	35	$2^{-195}$
5	254	253	153	

**Fig. 1.** Heuristical rates of failure to find an equivalent ideal of the desired norm type. For NIST level 3 this is formula (4). For the other NIST levels the formula was tweaked so as to take into account Remark 4.

### 5.1 On the Distribution of Auxiliary Paths

Let  $\tau : E_0 \rightarrow E_A$  be an  $N_\tau$ -isogeny and  $I_\tau$  be the left  $\mathcal{O}_0$ -ideal corresponding to  $\tau$ . Given the right order  $\mathcal{O}_A$  of  $I_\tau$ , we use  $\mathcal{S}_{I_\tau, M}$  to denote the distribution on

$$\mathcal{O}_M := \{\alpha \in \mathcal{O}_0 \cap \mathcal{O}_A \mid n(\alpha) = M\}$$

that are outputs of the algorithm consisting of first getting  $(C_0 : D_0) \in \mathbb{P}^1(\mathbb{Z}/N_\tau\mathbb{Z})$  by running `EichlerModConstraint`( $I_\tau, 1, 1$ ), then getting  $\alpha \in \mathcal{O}_0 \cap \mathcal{O}_A$  with norm  $M$  by running `StrongApproximationM`( $N_\tau, C_0, D_0$ ).

For a fixed  $q$ , we define

$$\text{Iso}(E_A, q) := \{\varphi : E_A \rightarrow \star \text{ such that } \deg \varphi = 2^a - q\},$$

and we consider the following distributions on  $\text{Iso}(E_A, q)$ :

- $\mathcal{D}_U$ : The uniform distribution  $\mathcal{U}_{\text{Iso}(E_A, q)}$ .
- $\mathcal{D}_1$ : For  $q$  such that  $d = q(2^a - q)$  satisfies Eq. (2): a factor of  $\theta_\alpha$  of degree  $2^a - q$  where  $\alpha \sim \mathcal{S}_{I_\tau, M(q)}$  and  $\theta_\alpha \in \text{End}(E_A)$  is the corresponding endomorphism.
- $\mathcal{D}_2$ : For  $q$  such that  $d = q(2^a - q)$  does not satisfy Eq. (2): a factor of  $\theta_\alpha \circ \theta''$  of degree  $2^a - q$  where  $\alpha \sim \mathcal{S}_{I_\tau, M(q)/3}$ ,  $\theta_\alpha \in \text{End}(E_A)$  is the corresponding endomorphism and  $\theta''$  is a random isogeny of degree 3 with domain  $E_A$ .
- $\mathcal{D}_{AP}$ :  $\mathcal{D}_{AP} = \mathcal{D}_1$  if  $d = q(2^a - q)$  satisfies Eq. (2), and  $\mathcal{D}_{AP} = \mathcal{D}_2$  otherwise. Note that this is the same distribution as the outputs of Algorithm 3 with  $d = q, D_1 = 2^a$  and  $D = 2^{a+b}$ .

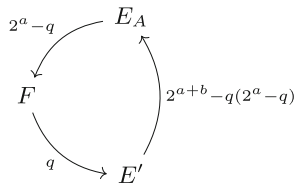
Finally, we define a distribution  $\mathcal{Q}$  on  $\mathbb{Z}$ , which is the distribution of reduced norm of the response ideals  $I_\sigma$ .

*Problem 1.* Let  $a$  be a fixed integer as in the parameter choices and  $E_A$  be the public curve. Let  $S = \{\omega : E_A \rightarrow \star \text{ of degree } 2^a - q\}$  be a set of size  $M > \log N_\tau$  where either

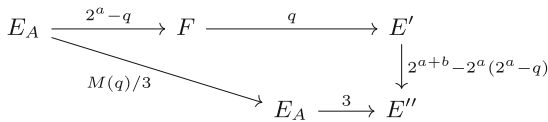
1.  $S$  is sampled by first sampling  $q \sim \mathcal{Q}$ , then sampling  $\omega$  from  $\mathcal{D}_U$ ;
2.  $S$  is sampled by first sampling  $q \sim \mathcal{Q}$ , then sampling  $\omega$  from  $\mathcal{D}_{AP}$ .

The problem is, given  $E_A, a, S$ , to distinguish between the two cases with a polynomial number of queries to  $\mathcal{Q}$ , FIDIO and to  $\mathcal{D}_{AP}$ .

*Remark 5.* It seems that the most natural way to distinguish the two cases in Problem 1 is to reverse engineer the algorithm that underlies the distribution  $\mathcal{D}_{\mathcal{AP}}$ . That means, given an isogeny  $E_A \rightarrow F$  of degree  $2^a - q$ , one tries to complete the diagrams in Figs. 2 and 3. In the first case, it means to come up with an isogeny from  $F$  to  $E_A$  of degree  $q(2^{a+b} - q(2^a - q))$ . This gives rise to an endomorphism on  $E_A$ , then one recovers the quaternion element corresponding to this endomorphism and check whether the quaternion element is sampled from  $\mathcal{S}_{I_\tau, M(q)}$ . The second case is similar, except that one finds an isogeny from  $F$  to some curve  $E''$  that is away from  $E_A$  by a degree 3 isogeny. This process, requires at least the knowledge of both the endomorphism rings of  $E_A$  and  $F$ . Therefore, it seems reasonable to assume that an algorithm to solve Problem 1 requires at least  $O(2^\lambda)$  time complexity.



**Fig. 2.** A diagram that illustrates the computation of the auxiliary path from  $E_A$  in the case when Eq. (2) holds for  $d = q(2^a - q)$  and  $D = 2^{a+b}$ .



**Fig. 3.** A diagram that illustrates the computation of the auxiliary path from  $E_A$  in the case when Eq. (2) does not hold.

**5.2 Soundness of SQIsign2D-East**

The proof of soundness of our protocol is quite similar to that of SQIsignHD. Let  $(E_1, \phi, K_g, E_4, U_2, V_2)$  and  $(E_1, \phi', K'_g, E'_4, U'_2, V'_2)$  are two SQIsign2D-East transcripts with the same commitment  $E_1$  but different challenges  $\phi \neq \phi'$ . From  $(K_g, E_4, U_2, V_2)$  and  $(K'_g, E'_4, U'_2, V'_2)$ , we can compute efficient representations of  $\sigma : E_A \rightarrow E_2$  and  $\sigma' : E_A \rightarrow E'_2$ , where  $E_2$  and  $E'_2$  are codomains of  $\phi$  and  $\phi'$ , respectively.

Therefore, we obtain an efficient representation of  $\alpha = \hat{\sigma}' \circ \phi' \circ \hat{\phi} \circ \sigma \in \text{End}(E_A)$ . Finally, the proof that  $\alpha$  is non-scalar is exactly same as SQIsignHD since it depends only on the fact that  $q = \text{deg}(\sigma)$  and  $q' = \text{deg}(\sigma')$  are coprime to  $\text{deg}(\phi) = \text{deg}(\phi')$ .

### 5.3 Zero-Knowledge of SQIsign2D-East

We now conclude this section with a proof of the zero-knowledge property of our SQIsign2D-East.

**Definition 3.** *Given parameters  $f$  and  $a$ , a random uniform nice degree isogeny oracle (RUNDIO) is an oracle taking as input a supersingular elliptic curve  $E$  defined over  $\mathbb{F}_{p^2}$  and returning an efficient representation of a random isogeny  $\sigma : E \rightarrow E'$  of  $(2^a, 2^b, f)_3$ -nice degree prime such that:*

- (i) *The distribution of  $E'$  is uniform in the supersingular isogeny graph.*
- (ii) *The conditional distribution of  $\sigma$  given  $E'$  is uniform among isogenies  $E \rightarrow E'$  of  $(2^a, 2^b, f)_3$ -nice degree.*

The existence of RUNDIO is based on the Heuristic assumption Sect. 4.4 applied to our choices of parameter sets.

**Definition 4.** *A fixed degree isogeny oracle (FIDIO) is an oracle taking as input a supersingular elliptic curve  $E$  defined over  $\mathbb{F}_{p^2}$  and an integer  $N$ , and outputs a uniformly random isogeny  $\varphi : E \rightarrow E'$  (in efficient representation) with domain  $E$  and degree  $N$ .*

**Theorem 2.** *Assuming that the commitment curve  $E_1$  is both computationally indistinguishable from an elliptic curve chosen uniformly at random in the supersingular isogeny graph, and the hardness of Problem 1. Then the SQIsign2D-East identification protocol is computationally honest-verifier zero-knowledge in the RUNDIO and FIDIO model.*

*In other words, there exists a polynomial time simulator  $\mathcal{S}$  with access to a RUNDIO and a FIDIO that produces random transcripts which are computationally indistinguishable from honest transcripts.*

*Proof.* A transcript of SQIsign2D-East consists of  $(E_1, \phi, K_g, E_4, U_2, V_2)$ , where  $E_1$  is a commitment,  $\phi$  is a challenge,  $(K_g, E_4, U_2, V_2)$  can be uniquely computed from a  $q$ -isogeny  $\sigma$  and a  $(2^a - q)$ -isogeny  $\omega$ . (See Algorithm 7 for detail.) The simulator proceeds as follows:

1. Call the RUNDIO on input  $E_A$  to get an isogeny  $\sigma' : E_A \rightarrow E'_2$  of  $(2^a, 2^b, f)_3$ -nice degree  $q$ .
2. Generate an isogeny  $\hat{\phi}' : E'_2 \rightarrow E'_1$  of degree  $2^b$  uniformly at random.
3. Call the FIDIO on input  $(E_A, 2^a - q)$ , resulting in the isogeny  $\omega' : E_A \rightarrow E'_3$ .
4. Compute  $(K'_g, E'_4, U'_2, V'_2)$  from  $(\sigma', \omega')$ .

Then the procedure above gives rise to a simulated transcript as  $(E'_1, \phi', K'_g, E'_4, U'_2, V'_2)$ .

Let  $(E_1, \phi, K_g, E_4, U_2, V_2)$  be a real transcript where  $(K_g, E_4, U_2, V_2)$  is computed from the response isogeny  $\sigma : E_A \rightarrow E_2$  of degree  $q$  and the auxiliary path  $\omega : E_A \rightarrow E_3$  of degree  $2^a - q$ . From the properties of the RUNDIO and FIDIO and the assumptions we made in the theorem, we can see that:



1. By the definition of the RUNDIO,  $E'_2$  is uniformly random in the super-singular isogeny graph. Since  $\hat{\phi}'$  is a uniformly random isogeny from  $E'_2$  of degree  $2^b$ , its codomain curve  $E'_1$  is also uniformly random in the graph. By assumption,  $E_1$  and  $E'_1$  are computationally indistinguishable.
2.  $\phi$  and  $\phi'$  follow the same distribution as they are generated the same way.
3. Conditional to  $E'_2$ ,  $\sigma'$  is uniformly random among isogenies between  $E_A$  and  $E'_2$  of  $(2^a, 2^b)_3$ -nice degree by the definition of RUNDIO. Conditional to  $E_2$ ,  $\sigma$  has the same distribution by construction.
4. Assuming the hardness of Problem 1, conditional to  $q$ ,  $\omega$  is computationally indistinguishable from a random isogeny of degree  $2^a - q$  from  $E_A$ .
5. Item 3,4 combined shows that  $(K_g, E_4, U_2, V_2)$  is computationally indistinguishable from  $(K'_g, E'_4, U'_2, V'_2)$  as the distributions of  $(\sigma, \omega)$  and  $(\sigma', \omega')$  are computationally indistinguishable.  $\square$

*Remark 6.* The assumption on the distribution of the commitment curve  $E_1$  made in Theorem 2 is about analyzing the distribution of the outputs of the algorithm `RandIsogImg` given the input norm size. This has been discussed in great detail in [26] where this algorithm was first introduced. Based on the discussions there, we believe this assumption is reasonable.

*The Previous Attack Strategy Does Not Apply.* To run the attack as in Sect. 4.3 on `SQIsign2D-East`, we need to be able to solve the following problem:

*Problem 2.* Let  $\omega : E_A \rightarrow \star$  of degree  $2^a - q$  where either

1.  $\omega$  is sampled from  $\mathcal{D}_1$ ,
2.  $\omega$  is sampled from  $\mathcal{D}_2$ .

The problem is, given  $E_A, \omega$ , to distinguish with success rate 1 between the two cases with a polynomial number of queries to  $\mathcal{D}_{\mathcal{AP}}$ .

We prove in Proposition 1 that Problem 2 is no easier than Problem 1 assuming that the best algorithm to solve Problem 1 has complexity  $O(2^{\lambda'})$  where  $\lambda' \geq \lambda$ . This seems a reasonable assumption as discussed in Remark 5, and a necessary condition to have our protocol achieve  $\lambda$ -bits security. Proposition 1 then implies that our assumption on the hardness of Problem 1 ensures the hardness of Problem 2, therefore we do not need to make an extra assumption on Problem 2. This agrees with our intuition that if Problem 2 were easy, then our `SQIsign2D-East` would not be zero-knowledge.

**Proposition 1.** *If solving Problem 1 requires  $O(2^{\lambda'})$  time complexity with  $\lambda' \geq \lambda$ , then solving Problem 2 requires at least  $O(2^{\lambda'})$  time complexity.*

*Proof.* We prove by contradiction. Suppose there is an algorithm  $\mathcal{A}$  that solves Problem 2 in  $O(2^{\lambda''})$  where  $\lambda'' < \lambda$ . Now in Problem 1, we are given with  $M$  samples with  $M > \log N_\tau$  such that they are either from  $\mathcal{D}_U$  or  $\mathcal{D}_{\mathcal{AP}}$ . We run the distinguishing algorithm  $\mathcal{A}$  on around  $\log N_\tau \approx \lambda/2$  number of samples to get enough Legendre symbol values with respect to  $N_\tau$  to uniquely determine

$N_\tau$ . These values allows us to recover  $N_\tau$  in time  $O(2^{\lambda/2})$ . Given the value of  $N_\tau$ , then we check whether the remaining  $M - \log N_\tau$  samples gives rise to correct Legendre symbols values. In the case when the  $M$  samples are from  $\mathcal{D}_U$ , this fails with a non-negligible probability; and in the case when  $M$  samples are from  $\mathcal{D}_{AP}$ , this always succeeds. This leads to an algorithm that solves Problem 1 in time  $\tilde{O}(2^{\lambda'} + 2^{\lambda/2})$  which is less than  $O(2^{\lambda'})$ , a contradiction.  $\square$

*Remark 7.* Although the additional 3-isogeny computation will probably be very fast if compared to the rest of the response step, it still introduces a conditional step that is performed only when  $q$  fails to satisfy some Legendre symbol condition with respect to  $N_\tau$ . This creates a side channel that may be exploited leading to a restoration of the original attack. We leave this solution as a future work.

## 6 Efficiency

In this section, we analyse the efficiency of SQIsign2D-East and CompactSQIsign2D-East. First, we provide concrete parameters for these protocols, then compare the data sizes of these protocols such as public key size and ciphertext size with SQIsign and SQIsignHD. Finally, we analyse the computational cost of SQIsign2D-East and CompactSQIsign2D-East.

### 6.1 Parameters

In the following, we give concrete parameters for SQIsign2D-East and CompactSQIsign2D-East satisfying the NIST security level 1, 3, and 5.

NIST level	$a$	$b$	$f$	$p$
1	127	126	27	$2^{253} \cdot 27 - 1$
3	191	189	35	$2^{380} \cdot 35 - 1$
5	254	253	153	$2^{507} \cdot 153 - 1$

*Remark 8.* To fit primes into 64-bit limbs, it preferable to use smaller primes such as:  $p = 2^{248} \cdot 5 - 1$ ,  $p = 2^{376} \cdot 65 - 1$ , and  $p = 2^{500} \cdot 27 - 1$  used in SQIsign2D-West [2]. However, if we choose such primes, the challenge length  $b$  becomes quite smaller than  $\lambda$ . (e.g.  $b = 123 < 128$  for Level 1.) Therefore, we need to extend the challenge length in some way. For example, if there exists a smooth integer  $c \mid (p - 1)$ , we can extend the challenge degree from  $2^b$  to  $2^b \cdot c$  by using an additional  $c$ -isogeny. This change requires to evaluate points of order  $c$  under  $\psi$ , which is computed by a 2-dimensional isogeny. In the gluing step of the theta algorithm by [12], we need to compute the x-coordinate of the sum of an evaluated point and a point of order 4. This requires the arithmetic on  $\mathbb{F}_{p^4}$ . We leave the efficient computation to future work.

## 6.2 Data Sizes

In this subsection, we compare the signature sizes of SQIsign, SQIsignHD, SQIsign-2D-East, and CompactSQIsign2D-East using the above parameters. Table 1 shows each signature size. Note that we do not give the signature size of SQIsignHD for the level 3 and 5 since sufficient information to evaluate the signature sizes are not given in [11].

**Table 1.** Signature size comparison

Security	Protocol	Signature (bytes)
Level 1	SQIsign	177
	SQIsignHD	109
	<b>SQIsign2D-East</b>	<b>182</b>
	<b>CompactSQIsign2D-East</b>	<b>150</b>
Level 3	SQIsign	263
	SQIsignHD	–
	<b>SQIsign2D-East</b>	<b>271</b>
	<b>CompactSQIsign2D-East</b>	<b>223</b>
Level 5	SQIsign	335
	SQIsignHD	–
	<b>SQIsign2D-East</b>	<b>359</b>
	<b>CompactSQIsign2D-East</b>	<b>295</b>

As shown in Table 1, the signature size of SQIsign2D-East is larger than both SQIsign and SQIsignHD for every security level. On the other hand, the signature size of CompactSQIsign2D-East is smaller than SQIsign and larger than SQIsignHD for every security level.

## 6.3 Computational Cost

We compare the computational costs of SQIsignHD, SQIsign2D-East, and CompactSQIsign2D-East for the security level 1. Table 2 shows the number of isogeny computations of each degree. As Table 2 shows, our protocol does not require any 4-dimensional isogeny computation for the verification. In addition, the number of 2-dimensional isogeny computations is smaller than the number of 4-dimensional isogeny computations in SQIsignHD. Therefore, the verification cost of our protocol is clearly smaller than that of SQIsignHD. As for the key generation and signing, our protocol requires 2-dimensional isogeny computations, whereas SQIsignHD only requires 1-dimensional isogeny computations. Therefore, our protocol is likely to have a larger cost for the key generation and signing.

Finally, in Table 3, we show the actual computational times of SQIsign2D-East and CompactSQIsign2D-East implemented in Julia. The implementation is available as supplementary materials. These are the averages of 100 run times.

**Table 2.** Number of isogeny computations of each degree

Protocol (Security level 1)		2	3	(2, 2)	(2, 2, 2, 2)
SQIsignHD	keygen	378	234	–	–
	sign	252	312	–	–
	verify	–	78	–	142
SQIsign2D-East	keygen	–	–	253	–
	sign	126	0–4	633	–
	verify	126	0–4	127	–
CompactSQIsign2D-East	keygen	–	–	253	–
	sign	126	0–4	760	–
	verify	126	0–4	127	–

**Table 3.** Computational times (sec.)

Security	Protocol	keygen	sign	verify
Level 1	SQIsign2D-East	0.50	1.50	0.24
	CompactSQIsign2D-East	0.52	1.87	0.32
Level 3	SQIsign2D-East	1.02	2.91	0.51
	CompactSQIsign2D-East	1.03	3.21	0.56
Level 5	SQIsign2D-East	1.52	4.21	0.72
	CompactSQIsign2D-East	1.57	4.97	0.80

The computational times are measured on a computer with an Intel Core i7-10700K CPU@3.70 Hz without Turbo Boost. The cost evaluation through an optimized implementation is a future work.

## 7 Conclusion

In this paper, we introduce SQIsign2D-East, a new variant of SQIsignHD, which requires only 2-dimensional isogeny computations for the verification, while SQI-signHD requires 4-dimensional isogeny computations. As a building block of SQIsign2D-East, we construct a new algorithm, which is a generalization of the conventional algorithm called `RandIsogImg`. In addition, we propose CompactSQI-sign2D-East, which has shorter signature size but has larger signing cost.

Both SQIsign2D-East and CompactSQIsign2D-East have less verification costs than SQIsignHD. On the other hand, the signing costs are expected to be larger than SQIsignHD though they are expected to be smaller than SQIsign. The signature size of SQIsign2D-East is longer than both SQIsign and SQIsignHD. The signature size of CompactSQIsign2D-East is shorter than SQIsign but longer than SQIsignHD.

**Acknowledgments.** We would like to thank Andrea Basso, Luca De Feo, Pierrick Dartois, Antonin Leroux, Luciano Maino, and Benjamin Wesolowski for sharing their work on SQISign2D-West, and Max Duparc and Tako Boris Fouotsa for sharing their work on SQIPrime with us prior to publication. We also thank the anonymous ASIACRYPT 2024 reviewers for their valuable and constructive feedbacks.

## References

1. Reza Azarderakhsh, Matthew Campagna, Craig Costello, Luca De Feo, Basil Hess, Amir Jalali, David Jao, Brian Koziel, Brian LaMacchia, Patrick Longa, et al. Supersingular isogeny key encapsulation. *Submission to the NIST Post-Quantum Standardization project*, 152:154–155, 2017.
2. Andrea Basso, Luca De Feo, Pierrick Dartois, Antonin Leroux, Luciano Maino, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. SQISign2D-West: the Fast, the Small, and the Safer. Cryptology ePrint Archive, Paper 2024/760, 2024. <https://eprint.iacr.org/2024/760>.
3. Andrea Basso, Luciano Maino, and Giacomo Pope. FESTA: Fast encryption from supersingular torsion attacks. In *ASIACRYPT 2023*, pages 98–126, 2023.
4. Daniel J Bernstein, Luca De Feo, Antonin Leroux, and Benjamin Smith. Faster computation of isogenies of large prime degree. *Open Book Series*, 4(1):39–55, 2020.
5. Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In *EUROCRYPT 2023*, pages 423–447, 2023.
6. Jorge Chavez-Saab, Maria Corte-Real Santos, Luca De Feo, Jonathan Komada Eriksen, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Michael Meyer, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Francisco Rodríguez Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQISign. Submission to NIST standardization of additional digital signature schemes. <https://sqisign.org>, 2023.
7. Mingjie Chen, Antonin Leroux, and Lorenz Panny. SCALLOP-HD: group action from 2-dimensional isogenies. In *PKC 2024*, pages 190–216. Springer, 2024.
8. Henri Cohen. *A course in computational algebraic number theory*, volume 138. Springer Science & Business Media, 2013.
9. Maria Corte-Real Santos, Jonathan Komada Eriksen, Michael Meyer, and Krijn Reijnders. Apréssqi: extra fast verification for sqisign using extension-field signing. In *EUROCRYPT 2024*, pages 63–93. Springer, 2024.
10. Romain Cosset and Damien Robert. Computing  $(l, l)$ -isogenies in polynomial time on Jacobians of genus 2 curves. *Mathematics of Computation*, 84(294):1953–1975, 2015.
11. Pierrick Dartois, Antonin Leroux, Damien Robert, and Benjamin Wesolowski. SQISignHD: new dimensions in cryptography. In *EUROCRYPT 2024*, pages 3–32. Springer, 2024.
12. Pierrick Dartois, Luciano Maino, Giacomo Pope, and Damien Robert. An Algorithmic Approach to  $(2, 2)$ -isogenies in the Theta Model and Applications to Isogeny-based Cryptography. Cryptology ePrint Archive, Paper 2023/1747, 2023. <https://eprint.iacr.org/2023/1747>.
13. Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In *ASIACRYPT 2020*, pages 64–93, 2020.

14. Luca De Feo, David Kohel, Antonin Leroux, Christophe Petit, and Benjamin Wesolowski. SQISign: Compact post-quantum signatures from quaternions and isogenies. In *Asiacrypt Vol. 1*, volume 12491 of *Lecture Notes of Computer Science*, pages 64–93. Springer, 2020.
15. Luca De Feo, Antonin Leroux, Patrick Longa, and Benjamin Wesolowski. New algorithms for the deuring correspondence: towards practical and secure sqisign signatures. In *EUROCRYPT 2023*, pages 659–690. Springer, 2023.
16. Max Deuring. Die typen der multiplikatorenringe elliptischer funktionenkörper. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 14:197–272, 1941.
17. Max Duparc and Tako Boris Fouotsa. SQIPrime: A dimension 2 variant of SQISignHD with non-smooth challenge isogenies. Cryptology ePrint Archive, Paper 2024/773, 2024. <https://eprint.iacr.org/2024/773>.
18. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO 1986*, pages 186–194. Springer, 1986.
19. Everett W. Howe, Franck Leprévost, and Bjorn Poonen. Large torsion subgroups of split Jacobians of curves of genus two or three. *Forum Mathematicum*, 12(3):315–364, 2000.
20. David Jao and Luca De Feo. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In *PQCrypto 2011*, pages 19–34, 2011.
21. Ernst Kani. The number of curves of genus two with elliptic differentials. *Journal für die reine und angewandte Mathematik*, 485:93–122, 1997.
22. Markus Kirschmer and John Voight. Algorithmic enumeration of ideal classes for quaternion orders. *SIAM Journal on Computing*, 39(5):1714–1747, 2010.
23. David Lubicz and Damien Robert. Computing isogenies between abelian varieties. *Compositio Mathematica*, 148(5):1483–1515, 2012.
24. Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery on SIDH. *EUROCRYPT 2023*, pages 448–471, 2023.
25. Tomoki Moriya. IS-CUBE: An isogeny-based compact KEM using a boxed SIDH diagram. Cryptology ePrint Archive, Paper 2023/1506, 2023. <https://eprint.iacr.org/2023/1506>.
26. Kohei Nakagawa and Hiroshi Onuki. QFESTA: Efficient algorithms and parameters for FESTA using quaternion algebras. In *Annual International Cryptology Conference*, pages 75–106. Springer, 2024.
27. Hiroshi Onuki and Kohei Nakagawa. Ideal-to-isogeny algorithm using 2-dimensional isogenies and its application to SQISign. Cryptology ePrint Archive, Paper 2024/778, 2024. <https://eprint.iacr.org/2024/778>.
28. Damien Robert. Breaking SIDH in polynomial time. In *EUROCRYPT 2023*, pages 472–503, 2023.
29. Maria Corte-Real Santos, Craig Costello, and Benjamin Smith. Efficient (3,3)-isogenies on fast kummer surfaces. Cryptology ePrint Archive, Paper 2024/144, 2024.
30. Benjamin Andrew Smith. *Explicit endomorphisms and correspondences*. Phd thesis, University of Sydney, 2005.
31. Lázlo Tóth. A survey of gcd-sum functions. *Journal of Integer Sequences*, 13:article 10.8.1, 2010.
32. Jacques Vélú. Isogénies entre courbes elliptiques. *Comptes-Rendus de l'Académie des Sciences*, 273:238–241, 1971.



# An Algorithmic Approach to $(2, 2)$ -Isogenies in the Theta Model and Applications to Isogeny-Based Cryptography

Pierrick Dartois<sup>1,2</sup> , Luciano Maino<sup>3</sup> , Giacomo Pope<sup>3,4</sup>,  
and Damien Robert<sup>1,2</sup> 

<sup>1</sup> Univ. Bordeaux, CNRS, Bordeaux INP, IMB, UMR 5251, 33400 Talence, France

<sup>2</sup> INRIA, IMB, UMR 5251, 33400 Talence, France

<sup>3</sup> University of Bristol, Bristol, UK

<sup>4</sup> NCC Group, Cheltenham, UK

**Abstract.** In this paper, we describe an algorithm to compute chains of  $(2, 2)$ -isogenies between products of elliptic curves in the theta model. The description of the algorithm is split into various subroutines to allow for a precise field operation count.

We present a constant time implementation of our algorithm in Rust and an alternative implementation in SageMath. Our work in SageMath runs ten times faster than a comparable implementation of an isogeny chain using the Richelot correspondence. The Rust implementation runs up to forty times faster than the equivalent isogeny in SageMath and has been designed to be portable for future research in higher-dimensional isogeny-based cryptography.

## 1 Introduction

The devastating attacks on SIDH [3, 25, 42] have highlighted the relevance of studying higher-dimensional abelian varieties in isogeny-based cryptography. Following the attacks, it soon became evident that these new tools would have applications beyond cryptanalysis. For instance, Robert leveraged these techniques both to give a representation of isogenies in polylogarithmic time [40] and to compute the endomorphism ring of ordinary elliptic curves in quantum polynomial time [41].

On a more cryptographic side, the attacks have been used to design new protocols. Basso, Maino and Pope utilise these cryptanalytic techniques to construct

---

Author list in alphabetical order; see <https://www.ams.org//profession/leaders/CultureStatement04.pdf>. The first and fourth authors have been supported by the Agence Nationale de la Recherche under grant ANR-19-CE48-0008 (CIAO) and the France 2030 program under grant ANR-22-PETQ-0008 (PQ-TLS). The second author has been supported by the UK Engineering and Physical Sciences Research Council (EPSRC) Centre for Doctoral Training (CDT) in Trust, Identity, Privacy and Security in Large-scale Infrastructures (TIPS-at-Scale) at the Universities of Bristol and Bath.

a trapdoor mechanism, and using standard transformations, this trapdoor is used to derive a public-key encryption protocol named FESTA [2]. Subsequently, three additional protocols employing similar ideas to FESTA have appeared [27, 28, 33]. One of the main building blocks underlying these protocols is the computation of chains of  $(2, 2)$ -isogenies between products of two elliptic curves. However, the cryptographic application of these isogeny chains extends beyond FESTA-based applications. For instance, SQISign2D-West [1] uses two-dimensional isogenies between elliptic products to achieve significant speed ups for keygen and signing as well as the fastest SQISign verification to date. These isogenies are also at the core of computing the group action in SCALLOP-HD [4], as well as in novel constructions of isogeny-based weak verifiable delay functions [15] and verifiable random functions [21]. Therefore, improving algorithms to compute chains of  $(2, 2)$ -isogenies between elliptic products is of paramount importance to the progress of higher-dimensional isogeny-based protocols.

Prior to this work, the only method to compute  $(2, 2)$ -isogenies between elliptic products relied on ad-hoc procedures for gluing and splitting, and the use of the Richelot correspondence to compute isogenies between Jacobians of genus-two hyperelliptic curves [34, 45]. This method can be considered satisfactory for cryptanalytic purposes, but it is definitely not efficient enough for constructive applications. Indeed, for the proof-of-concept implementations of [2, 33], the two-dimensional isogenies are the bottleneck of the protocol. Richelot isogenies describe  $(2, 2)$ -isogenies between Jacobians of genus-two hyperelliptic curves in the Mumford model. Here, kernel elements are divisors, represented by a pair of univariate polynomials. The arithmetic of the group elements, as well as isogeny codomain computation and evaluation, require working in a univariate polynomial ring above the base field. This model makes doubling and evaluation of points expensive and the implementation of the isogeny chain itself is significantly more complicated than the more familiar isogeny chains between two elliptic curves, which use Vélu's formulae. A natural question is then to ask whether it could be possible to use different models that are more amenable to simple and efficient algorithmic descriptions.

In the literature, another model used to compute isogenies is already known: the *theta model*. Despite being suitable for isogenies between elliptic curves, the theta model has mainly been employed to compute isogenies in higher dimension due to the lack of alternatives. For instance, Cosset and Robert describe an algorithm for  $(\ell, \ell)$ -isogenies in the theta model for odd primes  $\ell$  [9], later improved in [24]. The case  $\ell = 2$  has been briefly treated in [38, Proposition 6.3.5] and [39, Remarks 2.10.3, 2.10.7, 2.10.14] but never formalised.

The theta model is well known for its efficient arithmetic (in low dimension). For instance, Chudnovsky and Chudnovsky utilised the arithmetic of Kummer surfaces represented in the theta model for factoring integers [6]. Following this work, Gaudry derived fast formulae for the scalar multiplication on the Kummer surface associated to genus-two hyperelliptic curves. Moreover, in [37], Renes, Schwabe, Smith and Batina designed signature schemes for microcontrollers based on the efficient Montgomery ladder scalar multiplication on the Kummer surface.

Despite this efficient arithmetic, curiously, up until now, it had not really been considered for efficient  $(2, 2)$ -isogenies between Kummer surfaces. A notable



exception is the work of Costello [10]. Costello employs the theta model to translate the computation of  $2^n$ -isogenies between elliptic curves defined over  $\mathbb{F}_{p^2}$  to  $(2^n, 2^n)$ -isogenies between Kummer surfaces over  $\mathbb{F}_p$  via the Weil restriction. However, the  $(2, 2)$ -isogenies considered by Costello are of a very special form, i.e. those deriving from very special kernels on the Weil restriction of elliptic curves. Our work can be seen both as a specialisation of the results of [39] to the case most interesting for isogeny-based cryptography (namely  $(2^n, 2^n)$ -isogenies between product of elliptic curves or Kummer surfaces), and as a generalisation of [10] to general  $(2^n, 2^n)$ -isogenies.

In this work, we mainly focus on an algorithm to compute  $(2^n, 2^n)$ -isogenies between products of elliptic curves using the theta model. Since the applications we have in mind fall within the realm of isogeny-based cryptography, we specialise in the case of chains of  $(2, 2)$ -isogenies whose intermediate abelian surfaces are all Jacobians of genus-two hyperelliptic curves and the kernel generators are all rational. Indeed, in all the current schemes involving two-dimensional isogenies, encountering elliptic products in the middle of these chains occurs with negligible probability. Still, we briefly explain how to extend our algorithms to treat all cases in Appendix A.

Our aim is to demystify the hard algebraic geometry underpinning the theta model and make it accessible to cryptographers who want to employ isogenies between higher-dimensional abelian varieties within their protocols. The end result of our work is a set of concrete algorithms which describe the necessary pieces for computing isogenies between elliptic products; written to be particularly amenable to efficient and optimised implementations which are not all that different in appearance to the one-dimensional isogenies many are more familiar with.

## 1.1 Contributions

This paper has been written with the aim of being modular, using an algorithmic approach. All the formulae in the paper are mainly derived from the *duplication formula*. As a result, a reader uniquely interested in the computational results can assume the validity of the work in Sect. 2 and follow along the subsequent sections, which contain the explicit algorithms. From the duplication formula, we first re-obtain the addition formulae that have already been described in [17] and also give a precise operation count in the base field.

The algorithm to compute chains of  $(2, 2)$ -isogenies between elliptic products is split into various subroutines. Each subroutine is carefully described in algorithmic boxes; this allows for a precise field operation count. The main advantage of this approach is that both reducible and irreducible abelian surfaces can be described in the same way. However, some extra care will be devoted to the splitting and gluing case.

The gluing case is the most delicate one, where zero coordinates must be carefully handled during both arithmetic and isogeny computations. We efficiently compute the theta model representation of an elliptic product using only the dimension one representation of the theta structures and a few additional

**Table 1.** Base field costs of doubling, codomain computation and evaluation for generic (normalised and projective) and gluing isogenies in the theta model. We denote by  $\mathbf{M}, \mathbf{S}, \mathbf{I}$  the costs of multiplication, squaring and inversion of an element in the base field and ignore the cost of additions. Computation of a codomain is given along with the precomputation cost which accounts for the one-time cost of computing field elements used when doubling and evaluating theta points along the isogeny chain.

Isogeny Type	Doubling	Codomain		Evaluation
		Precomputations	Codomain	
Normalised	$8\mathbf{S} + 6\mathbf{M}$	$4\mathbf{S} + 24\mathbf{M} + \mathbf{I}$	$8\mathbf{S} + 10\mathbf{M} + \mathbf{I}$	$4\mathbf{S} + 3\mathbf{M}$
Projective	$8\mathbf{S} + 8\mathbf{M}$	$5\mathbf{S} + 14\mathbf{M}$	$8\mathbf{S} + 7\mathbf{M}$	$4\mathbf{S} + 4\mathbf{M}$
Gluing	$12\mathbf{S} + 12\mathbf{M}$	—	$8\mathbf{S} + 13\mathbf{M} + \mathbf{I}$	$8\mathbf{S} + 10\mathbf{M} + \mathbf{I}$

multiplications to recover the product structure. A summary of the costs of the algorithms described in this paper is shown in Table 1.

Note that unlike the case of the Richelot chain, which requires both a  $(2, 2)$ -gluing and  $(2, 2)$ -splitting isogeny, computing the elliptic product at the end of a chain of isogenies in the theta model is a case of simply converting from one model to another, which can be done efficiently.

Finally, we offer both a constant time implementation of the computation of an isogeny between elliptic products in the programming language Rust, as well as an alternative implementation for the computer algebra system SageMath [46]. Both are available at the following GitHub repository:

<https://github.com/ThetaIsogenies/two-isogenies>.

The Rust implementation has been written with cryptographic applications in mind and so has been built to run in constant time, with the appropriate finite field arithmetic and no secret-dependent conditional branching. It should also be easily portable to other projects in the future. The SageMath implementation has been designed to be a drop-in replacement for the work of [34]. As a result, all the protocols whose implementation relies on this work or the proof-of-concept of [2] can be upgraded to  $(2, 2)$ -isogenies in the theta model with minimal effort. As a use case, we show the benefit of these algorithms in FESTA in Sect. 5.3.

We give explicit timings of our implementations in Table 2. In SageMath, our implementation achieves a ten times speed up for the codomain computation and more than twenty times speed up for evaluation time compared to [34]. For characteristic of size 254 bits, the Rust code runs approximately forty times faster than the same algorithms written in SageMath, and more than two times as fast for very large characteristic (1293 bits). Concretely, on an Intel Core i7-9750H CPU with a clock-speed of 2.6 GHz with turbo-boost disabled, we compute an isogeny chain of length  $n = 208$  between elliptic products over  $\mathbb{F}_{p^2}$  with a 254 bit characteristic in only 2.13 ms.

**Roadmap.** In Sect. 2, we give a concise summary of the algebraic theory of theta functions. The most important parts of this section are the duplication formula and the algorithm to construct theta structures on elliptic products; the reader willing to accept these two main building blocks can skip this section

entirely. In Sect. 3, we derive addition formulae from the duplication formula. The isogeny formulae are described in Sect. 4. We discuss our implementation results in Sect. 5 and draw some conclusions in Sect. 6.

**Notation.** Throughout the paper,  $\mathbf{M}, \mathbf{S}, \mathbf{I}$  will represent the cost of multiplication, squaring and inversion of an element in the base field, respectively. In Sect. 2, we will introduce the Hadamard transform  $\mathcal{H}$ ; in dimension two,

$$\mathcal{H}(x, y, z, w) = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

We also define  $(\tilde{\theta}_{00}(P) : \tilde{\theta}_{10}(P) : \tilde{\theta}_{01}(P) : \tilde{\theta}_{11}(P)) = \mathcal{H}(\theta_{00}(P), \theta_{10}(P), \theta_{01}(P), \theta_{11}(P))$  to be the *dual coordinates* of  $P$ , and the  $\star$  operator:

$$(x, y, z, w) \star (x', y', z', w') = (xx', yy', zz', ww').$$

Another useful operator we will introduce in Sect. 3 is the squaring operator  $\mathcal{S}$ :

$$\mathcal{S}(x, y, z, w) = (x^2, y^2, z^2, w^2).$$

When computing the cost of inverting  $k$  elements, we will use *batched inversions*. Batched inversions allow us to invert  $k$  elements at a cost of  $3(k - 1)$  multiplications and only one inversion [26, §10.3.1]. We refer to our implementation for an explicit description of the algorithm.<sup>1</sup>

## 2 Preliminaries

We assume the reader has some familiarity with  $(N, N)$ -isogenies between principally polarised abelian surfaces; we refer to [25, §2] for a cryptographer-friendly introduction to the subject (see also [7, Ch. V, §8] for a general introduction). Before giving an explicit description of the algorithm used to compute  $(2^n, 2^n)$ -isogenies between products of two elliptic curves, we provide a concise and self-contained summary of the *algebraic theory of theta functions*. Using theta functions, it is possible to perform arithmetic on *principally polarised abelian varieties*. The reader willing to assume the validity of the *duplication formula* and the algorithm to construct theta structures on elliptic products can skip this section entirely and use Algorithm 2 as a black box.

For all the other readers, in what follows, we utilise the language of *Mumford’s theory* to provide a summary of the algebraic theory of theta functions [29–31]. We will first briefly recapitulate Mumford’s results, then recall the *duplication formula*, which will allow us to describe isogeny formulae between principally polarised abelian surfaces, and finally provide a formula for the change of basis in the case of a product of two elliptic curves, which is the case analysed in the paper.

<sup>1</sup> Python implementation of batched inversion.

### 2.1 Mumford’s Theory

In [44, §2.2], Robert and Sarkis provide a concrete treatment of Mumford’s theory in the case of elliptic curves. In this section, we describe Mumford’s theory for a principally polarised abelian variety  $(A, \lambda)$  of dimension  $g$  in a similar manner, working over an algebraically closed field  $k$  of characteristic different from two.

Assume that the principal polarisation  $\lambda$  is represented by a divisor  $\Theta$ , which we will further assume to be symmetric:  $[-1]^*\Theta = \Theta$  (we can always find such a representative). The principal polarisation is then given by  $\lambda = \Phi_\Theta : A \rightarrow \hat{A}, P \mapsto t_P^*\Theta - \Theta$ .<sup>2</sup>

We can then define the polarisation of level  $n$ :  $\lambda \circ [n] : P \mapsto t_P^*n\Theta - n\Theta$ . Its kernel coincides with  $A[n]$ . This means that, if  $P \in A[n]$ ,  $t_P^*n\Theta - n\Theta$  is a principal divisor. We will denote by  $g_P$  a function on  $A$  with this divisor; the function  $g_P$  is well defined up to multiplication by an invertible constant in  $k$ . This is a fundamental ingredient to introduce the *theta group*.

The theta group of level  $n$  is given by  $\mathcal{G}(n\Theta) = \{(P, g_P), P \in A[n]\}$ , with group law  $(P, g_P) \cdot (Q, g_Q) = (P + Q, g_P(\_)g_Q(P + \_))$ . Finally we have an (irreducible) action of  $\mathcal{G}(n\Theta)$  on  $\Gamma(n\Theta) = \{f \in k(A)^* \mid \text{div}(f) \geq -n\Theta\} \cup \{0\}$ , via  $(P, g_P) \cdot f = g_P(\_)f(\_+P)$ . We also have an operator  $\delta_{-1}$  on  $\mathcal{G}(n\Theta)$ :  $\delta_{-1}(P, g_P) = (-P, [-1]^*g_P)$ .

Let  $\mathcal{G}(n)$  be the Heisenberg group  $k^* \times K(n) \times \hat{K}(n)$  where  $K(n) = (\mathbb{Z}/n\mathbb{Z})^g$  and  $\hat{K}(n) = (\hat{\mathbb{Z}}/n\hat{\mathbb{Z}})^g$ , where the multiplication is given by

$$(\alpha, x, \chi) \cdot (\alpha', x', \chi') := (\alpha\alpha'\chi'(x), x + x', \chi \cdot \chi').$$

We have an operator  $\delta_{-1}$  on  $\mathcal{G}(n)$  given by  $\delta_{-1}(\alpha, x, \chi) = (\alpha, -x, 1/\chi)$ , and an irreducible action of  $\mathcal{G}(n)$  to  $V(n)$ , the vector space of functions  $(\mathbb{Z}/n\mathbb{Z})^g \rightarrow k$  generated by the Kronecker delta functions  $\delta_i : i \in (\mathbb{Z}/n\mathbb{Z})^g$ , via  $(\alpha, x, \chi) \cdot \delta_i = \alpha\chi(i)\delta_{x+i}$ .

For technical reasons, we will from now on assume that  $n$  is even. We will denote by  $\mathcal{L}$  the line bundle associated to  $n\Theta$ . A *symmetric theta structure*  $\Theta^\mathcal{L}$  of type  $n$  is an isomorphism  $\mathcal{G}(n) \rightarrow \mathcal{G}(n\Theta)$  that commutes with the action of  $\delta_{-1}$  and which induces the identity on the natural embedding of  $k^*$  in both groups. It induces an isomorphism  $\overline{\Theta}^\mathcal{L} : A[n] \rightarrow H(n) = K(n) \times \hat{K}(n)$ , which sends the Weil pairing  $e_{n\Theta}$  on  $A[n]$  to the pairing  $e_n$  on  $H(n)$  given by  $e_n((x_1, \chi_1), (x_2, \chi_2)) = \chi_2(x_1)/\chi_1(x_2)$ . In particular, the symmetric theta structure of level  $n$   $\Theta^\mathcal{L}$  induces a canonical symplectic basis of the  $n$ -torsion; and Mumford shows in [29] that conversely  $\Theta^\mathcal{L}$  is induced by a symplectic basis of the  $2n$ -torsion. We will say that these bases are *compatible* with  $\Theta^\mathcal{L}$ .

By uniqueness of the irreducible action of the Heisenberg group [29], the theta structure  $\Theta^\mathcal{L}$  induces an isomorphism  $\beta : \Gamma(A, n\Theta) \xrightarrow{\sim} V(n)$ , uniquely defined up to a scalar. Via  $\beta$ , it is possible to transfer the basis  $\delta_i : i \in (\mathbb{Z}/n\mathbb{Z})^g$  of  $V(n)$ , to a basis  $(\theta_i)_{i \in (\mathbb{Z}/n\mathbb{Z})^g}$  on  $\Gamma(A, n\Theta)$ ; the functions  $\theta_i$  are called *theta coordinates* of level  $n$ . Using theta coordinates, it is possible to represent abelian varieties via an embedding into the projective space [32, Ch. II, Theorem 1.3]. In particular, if  $n > 2$ , the abelian variety  $A$  can be completely described in the

<sup>2</sup> With  $\hat{A}$ , we denote the dual abelian variety of  $A$ .

projective space via the evaluation of theta coordinates at the identity using the *Riemann relations*; we call the projective point  $(\theta_i(0^A))_{i \in (\mathbb{Z}/n\mathbb{Z})^g}$  the *theta-null point*. Given a point  $P \in A$  and  $T \in A[n]$ , we can efficiently represent  $P + T$  in theta coordinates: if  $T$  corresponds to  $(s, \chi)$  via  $\overline{\Theta}^{\mathcal{L}}$ ,

$$(\theta_i(P + T))_i = (\chi(i)\theta_{i+s}(P))_i. \tag{1}$$

Let  $f: A \rightarrow B$  be an isogeny between abelian varieties, let  $\Theta_A, \Theta_B$  be two divisors inducing principal polarisations on  $A$  and  $B$  respectively. Suppose there exists an isomorphism  $\alpha: f^*\Theta_B \xrightarrow{\sim} n\Theta_A$ , we say that  $f$  is an  $n$ -isogeny. Then, one can prove that  $\ker(f) \subset A[n]$ . On the other hand, given  $K \subset A[n]$ , it is not generally true that the isogeny  $f': A \rightarrow B$  of kernel  $K$  generates an isomorphism between  $n\Theta_A$  and  $(f')^*\Theta'_B$  for some divisor  $\Theta'_B$  on  $B$ . By [29], this is exactly true when  $K$  is *maximal isotropic* for the Weil pairing  $e_n$  on  $A[n]$ <sup>3</sup>. We note that if we have a theta structure  $\Theta^{\mathcal{L}}$  on  $A$ , then the image of  $K(n) = (\mathbb{Z}/n\mathbb{Z})^g$  and  $\hat{K}(n) = (\hat{\mathbb{Z}}/n\hat{\mathbb{Z}})^g$  by  $\overline{\Theta}^{\mathcal{L}}$  are maximal isotropic subgroups of  $A[n]$ . If  $K$  is equal to the image of  $\hat{K}(n)$  by  $\overline{\Theta}^{\mathcal{L}}$ , we say that it is *compatible* with the theta structure.

In this work we will consider  $(2^n, 2^n)$ -isogenies  $A \rightarrow B$  between abelian surfaces, with kernel  $K$  maximal isotropic in  $A[2^n]$ , and  $A$  will be endowed with a symmetric theta structure of level two. We will say that  $K$  is *compatible with our theta structure* if not only  $K[2]$  is compatible in the sense above, but also that  $K[4]$  is compatible with the symmetric structure. This is to say that if  $T' \in K[4]$  is a point of exact order four, and  $T = 2T'$ , with  $T$  corresponding to  $\overline{\Theta}^{\mathcal{L}}(0, \chi)$ , then we require the theta coordinates  $\theta_i(T')$  to be invariant under the action of  $\Theta^{\mathcal{L}}(1, 0, \chi)$ . Unraveling the definition, this is equivalent to the fact that a basis of  $K[4]$  extends to a symplectic basis of  $A[4]$  compatible with  $\Theta^{\mathcal{L}}$ .

*Example 1.* Let  $(a : b : c : d)$  be a theta-null point of level two in dimension two obtained from the theta structure  $\Theta^{\mathcal{L}}$ . Implicitly, this determines a symplectic basis  $(S_1, S_2, T_1, T_2)$  of the two-torsion. Let  $i_1 = ([1], [0]) \in K(2)$ ,  $i_2 = ([1], [0]) \in K(2)$ ,  $\chi_1, \chi_2 \in \hat{K}(2, 2)$  such that

$$\chi_j(i_k) = \begin{cases} -1 & \text{if } j = k, \\ 1 & \text{if } j \neq k. \end{cases}$$

Then, let us define  $S_1 = \overline{\Theta}^{\mathcal{L}}(i_1)$ ,  $S_2 = \overline{\Theta}^{\mathcal{L}}(i_2)$ ,  $T_1 = \overline{\Theta}^{\mathcal{L}}(\chi_1)$  and  $T_2 = \overline{\Theta}^{\mathcal{L}}(\chi_2)$ . If  $P = (x : y : z : t)$ , we have  $S_1 = (b : a : d : c)$  and  $P + S_1 = (y : x : t : z)$ ,  $S_2 = (c : d : a : b)$  and  $P + S_2 = (z : t : x : y)$ ,  $T_1 = (a : -b : c : -d)$  and  $P + T_1 = (x : -y : z : -t)$ ,  $T_2 = (a : b : -c : -d)$  and  $P + T_2 = (x : y : -z : -t)$ .

*Remark 2 (Rational theta coordinates).* In practice, our base field  $k$  is not algebraically closed. However, if we assume that  $A[2n]$  is  $k$ -rational then the associated level- $n$  theta structure is also  $k$ -rational. Especially, the theta-null point

---

<sup>3</sup> We say  $K$  is isotropic for  $e_n$  when  $e_n(x, y) = 1$  for all  $x, y \in K$  and maximal isotropic if it is maximal as a subgroup of  $A[n]$  for this property.

is  $k$ -rational and level- $n$  theta coordinates of  $k$ -rational points are  $k$ -rational. In our work focused on cryptographic applications,  $k$  will be a finite field (e.g.  $\mathbb{F}_{p^2}$ ) but the formulae we provide are valid on any perfect field.

Another fundamental ingredient is the change of theta structure given by *Heisenberg group automorphisms*. A Heisenberg group automorphism is an automorphism of  $\mathcal{G}(n)$  acting as the identity on  $k^*$ . In particular, such an automorphism induces a symplectic automorphism on  $H(n)$  with respect to its natural pairing  $e_n$ . The most fundamental example is the *Hadamard Transform*, which is the automorphism that swaps  $K(n)$  and  $\hat{K}(n)$ .

**Hadamard Transform.** Let  $(\theta_i)_i$  be some theta coordinates on  $A$ . The action of the Hadamard transform on  $(\theta_i)_i$  is described in [39, Eq. 2.4]. The resulting theta coordinates after this transform are called the *dual theta coordinates*; we will denote such coordinates by  $(\tilde{\theta}_i)_i$ . In what follows, we will use the Hadamard transform on level-two theta coordinates. For the sake of clarity, we explicitly state the action of this symplectic automorphism in dimension one and two.

First, let us fix an ordering for theta coordinates. In dimension one, there are only two theta coordinates. Whenever we write  $(x : y)$  to represent a point  $P$  in theta coordinates, we actually mean  $(\theta_0(P) : \theta_1(P))$ . Hence, specialising [39, Eq. 2.4], we obtain  $(\tilde{\theta}_0(P) : \tilde{\theta}_1(P)) = (x + y : x - y)$ . In dimension two, we represent a point  $P$  in theta coordinates by a tuple  $(x : y : z : w)$ , where we fix the ordering  $(\theta_{00}(P) : \theta_{10}(P) : \theta_{01}(P) : \theta_{11}(P))$ .<sup>4</sup> Specialising [39, Eq. 2.4], we have

$$\begin{aligned} \tilde{\theta}_{00}(P) &= x + y + z + w, & \tilde{\theta}_{01}(P) &= x + y - z - w \\ \tilde{\theta}_{10}(P) &= x - y + z - w, & \tilde{\theta}_{11}(P) &= x - y - z + w. \end{aligned}$$

Henceforth, we will use the operator  $\mathcal{H}$  to refer to the action of the Hadamard transform on theta coordinates. Finally, we remark that  $\mathcal{H}(\mathcal{H}((\theta_i^A)_i)) = (\theta_i^A)_i$  (projectively).

### 2.2 Duplication Formula

Let  $(\theta_i^A)_i$  be theta coordinates of level two on  $A$ . Implicitly, we have a symplectic decomposition of  $A[2] = K(\mathcal{L}) = K(\mathcal{L})_1 \oplus K(\mathcal{L})_2$ —from now on, we will drop the dependence on the line bundle  $\mathcal{L}$  and simply write  $A[2] = K_1 \oplus K_2$ . Let  $(S_1, \dots, S_g)$  be the canonical basis induced by  $\overline{\Theta}^{\mathcal{L}}(K(2))$  and  $(T_1, \dots, T_g)$  the canonical basis induced by  $\overline{\Theta}^{\mathcal{L}}(\hat{K}(2))$ , which means  $K_1 = \langle S_1, \dots, S_g \rangle$  and  $K_2 = \langle T_1, \dots, T_g \rangle$ . Now, let us consider the isogeny  $f: A \rightarrow B$ , where  $\ker(f) = K_2$ . The abelian variety  $B$  is principally polarised and in turn can be endowed with a type two theta structure, whose theta coordinates are denoted by  $(\theta_i^B)_i$ . Also, let us define  $\star$  to be the operator such that  $(x_i)_i \star (y_i)_i = (x_i y_i)_i$ . Then, a consequence of the isogeny theorem [29, Theorem 4, p. 302] (see also [38, Theorem 3.6.4]) and duplication formula [29, Equation A, p. 332] shows that:

$$(\theta_i^A(P + Q))_i \star (\theta_i^A(P - Q))_i = \mathcal{H} \left( \left( (\tilde{\theta}_i^B(f(P)))_i \star (\tilde{\theta}_i^B(f(Q)))_i \right) \right), \quad (2)$$

<sup>4</sup> The subscript  $ij$  refers to the pair  $([i], [j]) \in K(2)$ .

$$\mathcal{H} \left( \left( \theta_i^A(\tilde{f}(R)) \right)_i \star \left( \theta_i^A(\tilde{f}(S)) \right)_i \right) = \left( \tilde{\theta}_i^B(R+S) \right)_i \star \left( \tilde{\theta}_i^B(R-S) \right)_i, \quad (3)$$

where  $\tilde{f}$  denotes the dual isogeny of  $f$  and  $(\tilde{\theta}_i^B)_i$  are the dual theta coordinates of  $(\theta_i^B)_i$ .

### 2.3 Theta Structures on Elliptic Products

We now focus on products of two elliptic curves and explain how to endow these products with a theta structure of type two. First, let us recall that every theta structure of type two comes from a symplectic basis of the four-torsion [29, Remark 4, p. 319]. Let  $E_1$  and  $E_2$  be two elliptic curves. The natural candidate for a theta structure on  $E_1 \times E_2$  is the *product theta structure*, which is obtained via the combination of the theta structures on elliptic curves [13, Lemma F.3.1].

**Proposition 3.** *Let  $(a_i : b_i)$  be theta-null points on  $E_i$  induced by a symplectic four-torsion basis  $(e_i, f_i)$ , for  $i = 1, 2$ . Then,  $(a_1a_2 : b_1a_2 : a_1b_2 : b_1b_2)$  is a theta-null point for  $E_1 \times E_2$  induced by the symplectic four-torsion  $\langle (e_1, 0), (0, e_2) \rangle \oplus \langle (f_1, 0), (0, f_2) \rangle$ .*

However, in the next sections, we might need to work with theta structures associated to a different symplectic four-torsion basis. In what follows, we explain how to construct theta structures associated with a fixed symplectic four-torsion basis. First, we explain how to do it over elliptic curves and then transfer our results to elliptic products.

Let  $E$  be an elliptic curve, and  $(T'_1, T'_2)$  a basis of the four-torsion. To compute the theta-null point associated to this basis, we proceed as follows. Given a point  $T \in E[2]$ , there are two *symmetric elements*  $\pm \mathfrak{g}$  (satisfying  $\delta_{-1}(\mathfrak{g}) = \mathfrak{g}^{-1}$ ) above  $T$  in the theta group  $\mathcal{G}(\mathcal{L}(2(0_E)))$ . We can fix a symmetric element via a point  $T'$  of four-torsion above  $T$ . Let  $T_1 = 2T'_1$ ,  $T_2 = 2T'_2$ , and let  $\mathfrak{g}_1, \mathfrak{g}_2$  be these elements associated to  $T'_1$  and  $T'_2$ , respectively. Unraveling the construction by Mumford of a symmetric theta structure of level two induced by a symplectic basis of level four, the theta coordinate  $\theta_0$  must be invariant under the action of  $\mathfrak{g}_2$ , and  $\theta_1 = \mathfrak{g}_1 \cdot \theta_0$ . The coordinate  $\theta_0$  can be computed as the trace of a global section  $s \in \Gamma(E, \mathcal{L}(2(0_E)))$ , provided it is not equal to zero, i.e.  $\theta_0 = \text{id} \cdot s + \mathfrak{g}_2 \cdot s \neq 0$ .

Working on a Montgomery curve in Weierstrass coordinates, we have a canonical point of four-torsion  $T' = (1 : 1)$  above  $T = (0 : 1)$  that induces the canonical element  $\mathfrak{g}$  of the theta group acting by  $\mathfrak{g} \cdot (X, Z) = (Z, X)$ . Indeed, translation by  $T$  is given by  $(X : Z) \mapsto (Z : X)$ , and the two symmetric elements above this translation act by  $(X, Z) \mapsto (\pm Z, \pm X)$  since they have order two. The element  $\mathfrak{g}_{T'}$  in  $\mathcal{G}(\mathcal{L}(2(0_E)))$  fixed by  $T'$  corresponds to  $\pm \mathfrak{g}$ . Still by unraveling Mumford’s construction, the correct sign choice for the symmetric element  $\mathfrak{g}_{T'}$  induced by  $T'$  is given by the one that leaves invariant any affine lift of  $T'$ . In our case, this is  $\mathfrak{g}$ .

For a general elliptic curve, if  $T' = (x, y, z)$  is a point of four-torsion and  $2T' = T = (u, v, w)$ , we can map  $T'$  to the Montgomery point  $(1 : 1)$  via the linear transformation (in the Kummer line):  $M : (X : Z) \mapsto (X' : Z') =$

$(zwX - zuZ : (xw - zu)Z)$ . It follows that the action of  $\mathfrak{g}_{T'}$  is given by

$$M^T U M^{T^{-1}} = \frac{1}{xw - zu} \begin{pmatrix} uz & zw \\ wx^2/z - 2ux & -uz \end{pmatrix},$$

with  $M = \begin{pmatrix} wz & -zu \\ 0 & xw - uz \end{pmatrix}$ ,  $U = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ . This computation is the output of Algorithm 1.

*Example 4.* Using previous notation, on a Montgomery curve we have  $T'_2 = (-1 : 1)$ , which acts by  $\mathfrak{g}_2 \cdot (X, Z) = (-Z, -X)$ . Taking the trace of  $X$  under this action we get  $\theta_0 = \text{id} \cdot X + \mathfrak{g}_2 \cdot X = X - Z$ .

Let  $T'_1 = (a + b : a - b)$  be another point of four-torsion; its double is then  $(a^2 + b^2 : a^2 - b^2)$ . Let  $x = a + b$ ,  $z = a - b$ ,  $u = a^2 + b^2$ ,  $w = a^2 - b^2$ . We compute  $\theta_1 = \mathfrak{g}_1 \cdot \theta_0 = \mathfrak{g}_1 \cdot (X - Z) = z(u - w)/(wx - uz)X + (wx^2/z - 2ux + uz)/(wx - uz)Z = b/aX + b/aZ$ . We recover the same conversion formula between Montgomery and theta coordinates as obtained in [43, Ch. 7, Appendix A.1].

We can use the same strategy to compute the theta-null point associated to a symplectic basis of the four-torsion on a product of elliptic curves. If  $T' = (T'_1, T'_2) \in E_1 \times E_2$  is a point of four-torsion, the associated element  $\mathfrak{g}_{T'}$  is given by  $\mathfrak{g}_{T'} = \mathfrak{g}_{T'_1} \otimes \mathfrak{g}_{T'_2}$ . Let  $X_i, Z_i$  be global sections of  $2(0_{E_i})$  defining the  $x$ -coordinate on  $E_i$  as  $x = X_i/Z_i$ . We can take  $\theta_0$  as the trace of  $X_1 \otimes X_2$ , i.e.  $\theta_0 = \sum_i \mathfrak{g}_i \cdot X_1 \otimes X_2 = \sum_i \mathfrak{g}_{i,1} \cdot X_1 \otimes \mathfrak{g}_{i,2} \cdot X_2$ , where the  $\mathfrak{g}_i = \mathfrak{g}_{i,1} \otimes \mathfrak{g}_{i,2}$ 's are the elements above  $K_2$  fixed by the four-torsion. The other theta coordinates are computed via the action of the elements above  $K_1$  on  $\theta_0$ .

---

**Algorithm 1.** Action by Translation

---

**Input:** A point  $P'$  in the four-torsion of the Kummer line of an elliptic curve

**Output:** The  $2 \times 2$  submatrix  $M$  with coefficients  $m_{ij}$  describing the action of  $\mathfrak{g}_{P'}$  lifting the action by translation of  $P = 2P'$ .

- 1:  $P \leftarrow [2]P'$  (▷) Cost: 2S + 3M
  - 2: Let  $P' = (X : Z)$  and  $(U : W) = P$
  - 3:  $WX, WZ, UX, UZ \leftarrow W \cdot X, W \cdot Z, U \cdot X, U \cdot Z$
  - 4:  $\delta \leftarrow WX - UZ$
  - 5: Compute  $\delta^{-1}, Z^{-1}$  via batched inversions (▷) Cost: 3M + 1I
  - 6:  $m_{00} \leftarrow -UZ \cdot \delta^{-1}$
  - 7:  $m_{01} \leftarrow -WZ \cdot \delta^{-1}$
  - 8:  $m_{10} \leftarrow UX \cdot \delta^{-1} - X \cdot Z^{-1}$
  - 9:  $m_{11} \leftarrow -m_{00}$
  - 10: **return**  $M$  (▷) Total cost: 2S + 14M + 1I
- 

In practice, in the algorithm to compute the  $(2^n, 2^n)$ -isogeny  $f : E_1 \times E_2 \rightarrow E'_1 \times E'_2$ , we only have access to  $\ker(f)[4] = \langle T'_1, T'_2 \rangle$  and not to a complete symplectic torsion basis of  $(E_1 \times E_2)[4]$ ; let  $T'_1 = (P_1, P_2)$  and  $T'_2 = (Q_1, Q_2)$ .



To bypass this problem, we define  $S'_1 = (0, Q_2)$  and  $S'_2 = (P_1, 0)$ .<sup>5</sup> Then,  $K_1 = \langle S_1, S_2 \rangle$  and  $K_2 = \langle T_1, T_2 \rangle$ , where  $S_i = [2]S'_i$  and  $T_i = [2]T'_i$ . We use the symplectic four-torsion basis  $(S'_1, S'_2, T'_1, T'_2)$  when endowing  $E_1 \times E_2$  with a theta structure. We summarise this procedure in Algorithm 2. The output of this algorithm is a matrix  $N$  that allows for a change of coordinates as follows. If  $R = (R_1, R_2) \in E_1 \times E_2$  is a point in Weierstrass coordinates for the Montgomery elliptic curves, where  $R_i = (X_i : Z_i)$ , the image of  $R$  in theta coordinates is given by  $N \cdot (X_1 \cdot X_2, X_1 \cdot Z_2, Z_1 \cdot X_2, Z_1 \cdot Z_2)^\top$ .

---

**Algorithm 2.** Change of Basis

---

**Input:** The points  $(P'_1, P'_2)$  and  $(Q'_1, Q'_2)$  in the four-torsion of  $E_1 \times E_2$  below the kernel.

**Output:** The  $4 \times 4$  change of basis matrix  $N$ .

- 1:  $G_1 \leftarrow \text{action\_by\_translation}(P'_1)$  (▷) Algorithm 1: Cost: 2S + 14M + 1I
  - 2:  $G_2 \leftarrow \text{action\_by\_translation}(P'_2)$
  - 3:  $H_1 \leftarrow \text{action\_by\_translation}(Q'_1)$
  - 4:  $H_2 \leftarrow \text{action\_by\_translation}(Q'_2)$
  - 5:  $t_{00|1} \leftarrow g_{00|1} \cdot h_{00|1} + g_{01|1} \cdot h_{10|1}$  (▷) Compute the first column of  $G_1 \times H_1$
  - 6:  $t_{10|1} \leftarrow g_{10|1} \cdot h_{00|1} + g_{11|1} \cdot h_{10|1}$
  - 7:  $t_{00|2} \leftarrow g_{00|2} \cdot h_{00|2} + g_{01|2} \cdot h_{10|2}$  (▷) Compute the first column of  $G_2 \times H_2$
  - 8:  $t_{10|2} \leftarrow g_{10|2} \cdot h_{00|2} + g_{11|2} \cdot h_{10|2}$
  - 9:  $n_{00} \leftarrow g_{00|1} \cdot g_{00|2} + h_{00|1} \cdot h_{00|2} + t_{00|1} \cdot t_{00|2} + 1$  (▷) Compute the trace for the first row
  - 10:  $n_{01} \leftarrow g_{00|1} \cdot g_{10|2} + h_{00|1} \cdot h_{10|2} + t_{00|1} \cdot t_{10|2}$
  - 11:  $n_{02} \leftarrow g_{10|1} \cdot g_{00|2} + h_{10|1} \cdot h_{00|2} + t_{10|1} \cdot t_{00|2}$
  - 12:  $n_{03} \leftarrow g_{10|1} \cdot g_{10|2} + h_{10|1} \cdot h_{10|2} + t_{10|1} \cdot t_{10|2}$
  - 13:  $n_{10} \leftarrow h_{00|2} \cdot n_{00} + h_{01|2} \cdot n_{01}$  (▷) Compute the action of  $(0, Q'_2)$  for the second row
  - 14:  $n_{11} \leftarrow h_{10|2} \cdot n_{00} + h_{11|2} \cdot n_{01}$
  - 15:  $n_{12} \leftarrow h_{00|2} \cdot n_{02} + h_{01|2} \cdot n_{03}$
  - 16:  $n_{13} \leftarrow h_{10|2} \cdot n_{02} + h_{11|2} \cdot n_{03}$
  - 17:  $n_{20} \leftarrow g_{00|1} \cdot n_{00} + g_{01|1} \cdot n_{02}$  (▷) Compute the action of  $(P'_1, 0)$  for the third row
  - 18:  $n_{21} \leftarrow g_{00|1} \cdot n_{01} + g_{01|1} \cdot n_{03}$
  - 19:  $n_{22} \leftarrow g_{10|1} \cdot n_{00} + g_{11|1} \cdot n_{02}$
  - 20:  $n_{23} \leftarrow g_{10|1} \cdot n_{01} + g_{11|1} \cdot n_{03}$
  - 21:  $n_{30} \leftarrow g_{00|1} \cdot n_{10} + g_{01|1} \cdot n_{12}$  (▷) Compute the action of  $(P'_1, Q'_2)$  for the final row
  - 22:  $n_{31} \leftarrow g_{00|1} \cdot n_{11} + g_{01|1} \cdot n_{13}$
  - 23:  $n_{32} \leftarrow g_{10|1} \cdot n_{10} + g_{11|1} \cdot n_{12}$
  - 24:  $n_{33} \leftarrow g_{10|1} \cdot n_{11} + g_{11|1} \cdot n_{13}$
  - 25: **return**  $N$  (▷) Total cost: 8S + 100M + 4I
- 

*Remark 5.* In Algorithm 2, it is possible to optimise the computation of the four inversions required in the four calls to Algorithm 1 in lines 1, 2, 3 and 4 by using a unique batched inversion. We decided not to show this optimisation in Algorithm 2 for the sake of a cleaner exposition.

---

<sup>5</sup> Here, we assume that both  $P_1$  and  $Q_2$  have order four. When this is not the case,  $f$  is a diagonal isogeny  $(P, Q) \mapsto (\phi_1(P), \phi_2(Q))$ , which can be computed via two one-dimensional isogenies  $\phi_1$  and  $\phi_2$ ; see also Appendix A.

### 3 Addition Formulae

In this section, we derive addition formulae using the equations in Sect. 2.2. These formulae have already been described in dimension two [17]. However, we prefer to restate them in dimension two to highlight the connection with (2, 2)-isogenies and provide an explicit operation count. In what follows, we use the same notation as in Sect. 2.2.

Let  $P, Q \in A$  and suppose we have  $(\theta_i^A(P - Q))_i$ . To compute  $(\theta_i^A(P + Q))_i$ , we can use Eq. 2, but first we need to recover  $(\tilde{\theta}_i^B(f(P)))_i$  and  $(\tilde{\theta}_i^B(f(Q)))_i$ , which can be computed as

$$(\tilde{\theta}_i^B(f(P)))_i \star (\tilde{\theta}_i^B(0))_i = \mathcal{H}((\theta_i^A(P))_i \star (\theta_i^A(P))_i),$$

and similarly for  $(\tilde{\theta}_i^B(f(Q)))_i$ . The quantity  $(\tilde{\theta}_i^B(0))_i$  is actually not needed, as we only need  $(\tilde{\theta}_i^B(0))_i \star (\tilde{\theta}_i^B(0))_i$  if we use

$$(\tilde{\theta}_i^B(f(P)))_i \star (\tilde{\theta}_i^B(0))_i \star (\tilde{\theta}_i^B(f(Q)))_i \star (\tilde{\theta}_i^B(0))_i,$$

to compute  $(\tilde{\theta}_i^B(f(P)))_i \star (\tilde{\theta}_i^B(f(Q)))_i$ . For the sake of compactness, we introduce the operator  $\mathcal{S}$  that, on input  $(\theta_i^A(P))_i$ , returns  $(\theta_i^A(P))_i \star (\theta_i^A(P))_i$ . We formalise this procedure in Algorithm 3.

Let  $(a : b : c : d)$  be a theta-null point for  $A$  and define  $(\alpha : \beta : \gamma : \delta)$  to be the dual coordinates  $(\tilde{\theta}_i^B(0))_i$  of the theta-null point  $(\theta^B(0))_i$ . For simplicity, let us assume that  $\alpha \cdot \beta \cdot \gamma \cdot \delta \neq 0$ ; the (rare) case when one of the dual coordinates is zero is briefly treated in Remark 6. Equation 2 proves that  $(\alpha^2 : \beta^2 : \gamma^2 : \delta^2) = \mathcal{H}(a^2 : b^2 : c^2 : d^2)$ . However, since we are working projectively, we need to use the quantities  $(\alpha^2/\beta^2, \alpha^2/\gamma^2, \alpha^2/\delta^2)$  and  $(a/b, a/c, a/d)$ , which can be computed via batched inversions.

To obtain an algorithm to double a point  $P \in A$ , we proceed as before with the only difference that we only need  $(a/b, a/c, a/d)$ . We provide a detailed description of the doubling in Algorithm 4.

*Remark 6.* In practice, we will always be in the case that  $\alpha \cdot \beta \cdot \gamma \cdot \delta \neq 0$ . We may incur in such an exception when we are working on a product of elliptic curves but with a non-product theta structure (see [43, Ch. 7, § 16.4]): this is due to how we construct the theta structure on the elliptic product in Sect. 2.3. In this case we do not have to worry since we could perform arithmetic on the elliptic curves and then convert to the theta model afterwards. However, if one wants to deal with this case in the theta model, one may first act with a symplectic automorphism  $\psi$  sending the dual theta-null point to an all-non-zero one, use the addition formulae and eventually switch back to the former theta structure acting by  $\psi^{-1}$ .

To compute  $(2^n, 2^n)$ -isogenies, we will only use doublings. Algorithm 4 works only if  $a \cdot b \cdot c \cdot d \neq 0$ . The case  $a \cdot b \cdot c \cdot d = 0$  can happen only if the codomain

**Algorithm 3.** Differential addition

**Input:** The theta coordinates of  $P$ ,  $Q$  and  $P - Q$ , and  $(\tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\lambda}_3) = (\alpha^2/\beta^2, \alpha^2/\gamma^2, \alpha^2/\delta^2)$ .

**Output:** The theta coordinates  $P + Q$ .

- 1:  $X_P, Y_P, Z_P, W_P \leftarrow \mathcal{H} \circ \mathcal{S}(x_P, y_P, z_P, w_P)$  (▷) Cost: 4S
- 2:  $X_Q, Y_Q, Z_Q, W_Q \leftarrow \mathcal{H} \circ \mathcal{S}(x_Q, y_Q, z_Q, w_Q)$  (▷) Cost: 4S
- 3:  $X_{f(P)f(Q)} \leftarrow X_P \cdot X_Q$
- 4:  $Y_{f(P)f(Q)} \leftarrow \tilde{\lambda}_1 \cdot Y_P \cdot Y_Q$
- 5:  $Z_{f(P)f(Q)} \leftarrow \tilde{\lambda}_2 \cdot Z_P \cdot Z_Q$
- 6:  $W_{f(P)f(Q)} \leftarrow \tilde{\lambda}_3 \cdot W_P \cdot W_Q$
- 7:  $X_{PQ}, Y_{PQ}, Z_{PQ}, W_{PQ} \leftarrow \mathcal{H}(X_{f(P)f(Q)}, Y_{f(P)f(Q)}, Z_{f(P)f(Q)}, W_{f(P)f(Q)})$
- 8:  $xy_{P-Q} \leftarrow x_{P-Q} \cdot y_{P-Q}$
- 9:  $zt_{P-Q} \leftarrow z_{P-Q} \cdot t_{P-Q}$
- 10:  $x_{P+Q} \leftarrow X_{PQ} \cdot zt_{P-Q} \cdot y_{P-Q}$
- 11:  $y_{P+Q} \leftarrow Y_{PQ} \cdot zt_{P-Q} \cdot x_{P-Q}$
- 12:  $z_{P+Q} \leftarrow Z_{PQ} \cdot xy_{P-Q} \cdot w_{P-Q}$
- 13:  $w_{P+Q} \leftarrow W_{PQ} \cdot xy_{P-Q} \cdot z_{P-Q}$
- 14: **return**  $x_{P+Q}, y_{P+Q}, z_{P+Q}, w_{P+Q}$  (▷) Total cost: 8S + 17M

**Algorithm 4.** Doubling

**Input:** The theta coordinates of  $P$  and  $(\tilde{\lambda}_1, \tilde{\lambda}_2, \tilde{\lambda}_3) = (\alpha^2/\beta^2, \alpha^2/\gamma^2, \alpha^2/\delta^2)$  and  $(\lambda_1, \lambda_2, \lambda_3) = (a/b, a/c, a/d)$ .

**Output:** The theta coordinates  $[2]P$ .

- 1:  $X_P, Y_P, Z_P, W_P \leftarrow \mathcal{H} \circ \mathcal{S}(x_P, y_P, z_P, w_P)$  (▷) Cost: 4S
- 2:  $X'_{f(P)}, Y'_{f(P)}, Z'_{f(P)}, W'_{f(P)} \leftarrow \mathcal{S}(X_P, Y_P, Z_P, W_P)$  (▷) Cost: 4S
- 3:  $Y'_{f(P)} \leftarrow \lambda_1 \cdot Y'_{f(P)}$
- 4:  $Z'_{f(P)} \leftarrow \tilde{\lambda}_2 \cdot Z'_{f(P)}$
- 5:  $W'_{f(P)} \leftarrow \tilde{\lambda}_3 \cdot W'_{f(P)}$
- 6:  $X'_P, Y'_P, Z'_P, W'_P \leftarrow \mathcal{H}(X'_{f(P)}, Y'_{f(P)}, Z'_{f(P)}, W'_{f(P)})$
- 7:  $Y'_P, Z'_P, W'_P \leftarrow \lambda_1 \cdot Y'_P, \lambda_2 \cdot Z'_P, \lambda_3 \cdot W'_P$
- 8: **return**  $X'_P, Y'_P, Z'_P, W'_P$  (▷) Total cost: 8S + 6M

$B$  is a product of elliptic curves with non product theta structure by [43, Ch. 7, § 16.4]. In this case, we can use the same solution as above, by using  $\psi = \mathcal{H}$  as our symplectic transformation.

## 4 The Isogeny Formula

In this section, we explain how to derive an isogeny formula for  $(2, 2)$ -isogenies from Sect. 2.2. Ultimately, we focus on chains of isogenies between products of two elliptic curves. However, we first show how to compute isogenies when we have an abelian surface already endowed with a theta structure compatible with the kernel of the  $(2, 2)$ -isogeny we want to compute.

Let  $A$  be an abelian surface defined over a perfect field  $k$  endowed with a  $k$ -rational theta structure of level two, and let  $(S_1, S_2, T_1, T_2)$  be the canonical

symplectic basis associated with the symplectic decomposition  $A[2] = K_1 \oplus K_2$ ; to be more specific,  $K_1 = \langle S_1, S_2 \rangle$  and  $K_2 = \langle T_1, T_2 \rangle$ . Let us recall that a theta-null point is fixed by a  $k$ -rational symplectic basis of the four-torsion [29, Remark 4, p. 319], and let  $(S'_1, S'_2, T'_1, T'_2)$  be such a basis; in particular  $2S'_i = S_i$  and  $2T'_i = T_i$ . Our goal is to compute a (2, 2)-isogeny  $f : A \rightarrow B$ . As explained in Sect. 2.3, in our case, we will always be working with  $\ker(f) = K_2$ .

Before outlining the explicit procedure, let us assume that we have  $k$ -rational points  $T''_1, T''_2$  such that  $\langle T''_1, T''_2 \rangle[4] = \langle T'_1, T'_2 \rangle$ ,  $2T''_i = T'_i$  and their Weil pairing  $e_8(T''_1, T''_2) = 1$ .<sup>6</sup> These conditions are not restrictive since they are naturally satisfied for chains of (2, 2)-isogenies, which are our end goal. In particular,  $(f(T''_1), f(T''_2))$  are two of the four-torsion points inducing the theta-null point on  $B$  lying above the two two-torsion points in  $K_2$ . Hence, the points  $(f(T''_1), f(T''_2))$  lay above the canonical points in  $K_1$  for the dual coordinates.

Let us remark that  $f(T''_i) + 2f(T''_i) = -f(T''_i)$ . So, as highlighted in Eq. 1 and since we are on the Kummer, we have

$$\begin{aligned} & \left( \tilde{\theta}_{00}^B(f(T''_1)) : \tilde{\theta}_{10}^B(f(T''_1)) : \tilde{\theta}_{01}^B(f(T''_1)) : \tilde{\theta}_{11}^B(f(T''_1)) \right) = \\ & \left( \tilde{\theta}_{10}^B(f(T''_1)) : \tilde{\theta}_{00}^B(f(T''_1)) : \tilde{\theta}_{11}^B(f(T''_1)) : \tilde{\theta}_{01}^B(f(T''_1)) \right), \end{aligned}$$

and

$$\begin{aligned} & \left( \tilde{\theta}_{00}^B(f(T''_2)) : \tilde{\theta}_{10}^B(f(T''_2)) : \tilde{\theta}_{01}^B(f(T''_2)) : \tilde{\theta}_{11}^B(f(T''_2)) \right) = \\ & \left( \tilde{\theta}_{01}^B(f(T''_2)) : \tilde{\theta}_{11}^B(f(T''_2)) : \tilde{\theta}_{00}^B(f(T''_2)) : \tilde{\theta}_{10}^B(f(T''_2)) \right). \end{aligned}$$

Define  $(\alpha : \beta : \gamma : \delta)$  to be the dual theta-null point of  $B$ , i.e.

$$\left( \tilde{\theta}_{00}^B(0) : \tilde{\theta}_{10}^B(0) : \tilde{\theta}_{01}^B(0) : \tilde{\theta}_{11}^B(0) \right) = (\alpha : \beta : \gamma : \delta).$$

Then, combining Eq. 2 with the above observations, we have

$$\begin{aligned} \mathcal{H} \circ \mathcal{S}(\theta_{00}^A(T''_1), \theta_{10}^A(T''_1), \theta_{01}^A(T''_1), \theta_{11}^A(T''_1)) &= (x\alpha, x\beta, y\gamma, y\delta), \\ \mathcal{H} \circ \mathcal{S}(\theta_{00}^A(T''_2), \theta_{10}^A(T''_2), \theta_{01}^A(T''_2), \theta_{11}^A(T''_2)) &= (z\alpha, w\beta, z\gamma, w\delta), \end{aligned}$$

for some unknown  $x, y, z, t$ . Hence, we can recover the dual theta-null point  $(\alpha : \beta : \gamma : \delta)$  for  $B$ , and in turn its theta-null point  $\mathcal{H}(\alpha : \beta : \gamma : \delta)$ .

*Remark 7 (Technical Remark).* It is possible to prove that all  $x, y, z, t$  must be different from zero. If it had not been the case, we would have ended up with a theta-null point with at least two zero coordinates. This is a contradiction since it implies we have more than a zero even theta-null coordinate of level (2, 2) – see Section “Gluing Isogeny” for the definition of level-(2, 2) theta coordinates.

In general, the dual theta-null point  $(\alpha : \beta : \gamma : \delta)$  has all coordinates different from zero. The only exceptions can be found for certain cases of the gluing isogeny – we will discuss how to handle this case in Sect. 4.1.

<sup>6</sup> Recall that such a subgroup  $\langle T''_1, T''_2 \rangle$  is said to be *isotropic*.

**Algorithm 5.** Codomain

**Input:** Theta coordinates of  $T_1''$  and  $T_2''$ , where  $T_i''$  is a 8-torsion point lying above the  $K_2$  part of the symplectic four-torsion basis inducing the theta-null point.

**Output:** Dual theta-null point  $(1 : \beta : \gamma : \delta)$ , the inverse of the dual theta-null point  $(1 : \beta^{-1} : \gamma^{-1} : \delta^{-1})$  and the theta-null point  $(a' : b' : c' : d')$  on  $B$ .

(▷) Case  $\beta \cdot \gamma \cdot \delta \neq 0$

- 1:  $(x\alpha, x\beta, y\gamma, y\delta) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_1''}, y_{T_1''}, z_{T_1''}, w_{T_1''})$  (▷) Cost: 4S
- 2:  $(z\alpha, w\beta, z\gamma, w\delta) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_2''}, y_{T_2''}, z_{T_2''}, w_{T_2''})$  (▷) Cost: 4S
- 3: Invert  $(x\alpha, x\beta, z\alpha, w\beta, z\gamma, w\delta)$  using batched inversions. (▷) Cost: 15M + 1I
- 4:  $\beta \leftarrow x\beta \cdot (x\alpha)^{-1}$
- 5:  $\gamma \leftarrow z\gamma \cdot (z\alpha)^{-1}$
- 6:  $\delta \leftarrow w\delta \cdot (w\beta)^{-1} \cdot \beta$
- 7:  $\beta^{-1} \leftarrow x\alpha \cdot (x\beta)^{-1}$
- 8:  $\gamma^{-1} \leftarrow z\alpha \cdot (z\gamma)^{-1}$
- 9:  $\delta^{-1} \leftarrow w\beta \cdot (w\delta)^{-1} \cdot \beta^{-1}$
- 10:  $(a', b', c', d') \leftarrow \mathcal{H}(1, \beta, \gamma, \delta)$
- 11: **return**  $(1, \beta, \gamma, \delta), (1, \beta^{-1}, \gamma^{-1}, \delta^{-1}), (a', b', c', d')$  (▷) Total cost: 8S + 10M + 1I + (13M)

Once we have computed  $(\alpha : \beta : \gamma : \delta)$ , we can evaluate the isogeny  $f$  at any point  $P$  using (again) Eq. 2. To compute the image, we first compute  $(x', y', z', w') = \mathcal{H} \circ \mathcal{S}((\theta_i^A(P))_i)$ . Then we find  $(\theta_i^B(f(P)))_i = \mathcal{H}(\alpha^{-1}x', \beta^{-1}y', \gamma^{-1}z', \delta^{-1}w')$  using  $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$  as input to the evaluation algorithm which can be computed at a one-time cost during the codomain computation.

**Algorithm 6.** Evaluation

**Input:** Theta coordinates of  $P$  and the dual theta-null point  $(1 : \beta^{-1} : \gamma^{-1} : \delta^{-1})$  on  $B$ .

**Output:** Theta coordinates  $f(P)$ . (▷) Case  $\beta \cdot \gamma \cdot \delta \neq 0$

- 1:  $(X_P, Y_P, Z_P, W_P) \leftarrow \mathcal{H} \circ \mathcal{S}(x_P, y_P, z_P, w_P)$  (▷) Cost: 4S
- 2:  $(X'_{f(P)}, Y'_{f(P)}, Z'_{f(P)}, W'_{f(P)}) \leftarrow (X_P, \beta^{-1} \cdot Y_P, \gamma^{-1} \cdot Z_P, \delta^{-1} \cdot W_P)$
- 3:  $(x_{f(P)}, y_{f(P)}, z_{f(P)}, w_{f(P)}) \leftarrow \mathcal{H}(X'_{f(P)}, Y'_{f(P)}, Z'_{f(P)}, W'_{f(P)})$
- 4: **return**  $(x_{f(P)}, y_{f(P)}, z_{f(P)}, w_{f(P)})$  (▷) Total cost: 4S + 3M

We give a detailed description of both the codomain and evaluation computations in Algorithms 5 and 6. We note that the cost in parentheses for Algorithm 5 is the cost of the computation of  $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$  which is required as input to Algorithm 6.

*Remark 8.* As we explained in Sect. 3, the inverse squared dual theta-null point  $(\alpha^2/\beta^2, \alpha^2/\gamma^2, \alpha^2/\delta^2)$  are also needed for the addition and doubling formulae. If such a quantity has already been precomputed at a cost of 4S + 15M + 1I, we can use it to lower down the cost in Algorithm 5. In line 3, we need only to invert three elements, namely  $x\alpha, z\alpha, w\beta$ . Then, to obtain  $(\beta^{-1}, \gamma^{-1}, \delta^{-1})$ ,

---

**Algorithm 7.** Projective Codomain

---

**Input:** Theta coordinates of  $T_1''$  and  $T_2''$ , where  $T_i''$  is a 8-torsion point lying above the  $K_2$  part of the symplectic four-torsion basis inducing the theta-null point.

**Output:** Dual theta-null point  $(\alpha : \beta : \gamma : \delta)$ , the inverse of the dual theta-null point  $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$  and the theta-null point  $(a' : b' : c' : d')$  on  $B$ .

- (▷) Case  $\alpha \cdot \beta \cdot \gamma \cdot \delta \neq 0$
- 1:  $(x\alpha, x\beta, y\gamma, y\delta) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_1''}, y_{T_1''}, z_{T_1''}, w_{T_1''})$  (▷) Cost: 4S
  - 2:  $(z\alpha, w\beta, z\gamma, w\delta) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_2''}, y_{T_2''}, z_{T_2''}, w_{T_2''})$  (▷) Cost: 4S
  - 3:  $z\alpha w\beta \leftarrow z\alpha \cdot w\beta$
  - 4:  $\alpha \leftarrow x\alpha \cdot z\alpha w\beta$
  - 5:  $\beta \leftarrow x\beta \cdot z\alpha w\beta$
  - 6:  $\gamma \leftarrow z\gamma \cdot x\alpha \cdot w\beta$
  - 7:  $\delta \leftarrow w\delta \cdot x\beta \cdot z\alpha$
  - 8:  $\alpha\beta, \gamma\delta \leftarrow \alpha \cdot \beta, \gamma \cdot \delta$
  - 9:  $\alpha^{-1} \leftarrow \gamma\delta \cdot \beta$
  - 10:  $\beta^{-1} \leftarrow \gamma\delta \cdot \alpha$
  - 11:  $\gamma^{-1} \leftarrow \alpha\beta \cdot \delta$
  - 12:  $\delta^{-1} \leftarrow \alpha\beta \cdot \gamma$
  - 13:  $(a', b', c', d') \leftarrow \mathcal{H}(\alpha, \beta, \gamma, \delta)$
  - 14: **return**  $(\alpha, \beta, \gamma, \delta), (\alpha^{-1}, \beta^{-1}, \gamma^{-1}, \delta^{-1}), (a', b', c', d')$  (▷) Total cost: 8S + 7M + (6M)
- 

we can simply multiply component-wise  $(\alpha^2/\beta^2, \alpha^2/\gamma^2, \alpha^2/\delta^2)$  by  $(\beta, \gamma, \delta)$ . The total cost in this optimised case is 8S + 10M + 1I + (4M).

**Projective Algorithms.** In Algorithm 5, we use batched inversions to recover  $(1 : \beta : \gamma : \delta)$  and  $(1 : \beta^{-1} : \gamma^{-1} : \delta^{-1})$ . This choice allows us to reduce the number of operations when doubling a point and evaluating an isogeny. However, we can remove the inversion in Algorithm 5 by working projectively at a cost of only a few extra multiplications. We describe a projective version of Algorithm 5 in Algorithm 7, where the cost in parentheses is associated to the computing the input  $(\alpha^{-1} : \beta^{-1} : \gamma^{-1} : \delta^{-1})$  used for the evaluation of theta points under the action of  $f$ .

Evaluating the isogeny and doubling a point when the dual theta-null point is not normalised at  $(1 : \beta : \gamma : \delta)$  induces an extra cost of one multiplication for evaluations and two multiplications for doubling. Additionally, the arithmetic precomputation requires 5S + 14M to precompute eight field elements. Note that if these arithmetic precomputations are available when computing the codomain, we can reduce the stated cost to 8S + 7M + (4M). Understanding whether to work projectively or using batched inversions with normalised null points boils down to the specifics of the chain length, number of evaluations and the cost of inversion in the base field. In the rest of paper, we will work using the normalised implementation for clarity but point out that it is always possible to use projective arithmetic to save inversions at the cost of slightly more expensive doubling and evaluations.

### 4.1 Computing $(2^n, 2^n)$ -Isogenies Between Elliptic Products

Now, we specialise to the case of a  $(2^n, 2^n)$ -isogeny  $f : E_1 \times E_2 \rightarrow E'_1 \times E'_2$  between elliptic products defined over a perfect field  $k$ , which will be computed as a chain of  $(2, 2)$ -isogenies. Let  $K$  be the kernel of this isogeny and suppose that we have two  $k$ -rational points of order  $2^{n+2}$  on  $E_1 \times E_2$  above  $K$  forming an isotropic group. To apply the formulae we described above, we need to be sure that  $K[4]$  is in “the right position”. Given  $K[4]$ , we apply Algorithm 2 to obtain a theta-null point induced by the symplectic four-torsion decomposition  $\langle S'_1, S'_2 \rangle \oplus \langle T'_1, T'_2 \rangle$ , where  $K[4] = \langle T'_1, T'_2 \rangle$

If  $n > 2$ , we have that  $K[8] = \langle T''_1, T''_2 \rangle$  is the isotropic 8-torsion above  $K[4]$ . This means we could apply Algorithms 5 and 6 to compute the first step of the isogeny  $f$ , i.e. the isogeny  $f_1 : E_1 \times E_2 \rightarrow A_1$  with kernel  $K[2]$ . However, we should be careful as on the product structure, one of the coordinates of the dual theta-null point on  $A_1$  may be equal to zero. We explain why this happens and how to bypass this obstacle in the Section “Gluing Isogeny” below.

After the first step, we also end up with a complete description of the theta structure on  $A_1$ ; let  $A_1[2] = K_1 \oplus K_2$ . The points  $f_1(T''_1)$  and  $f_1(T''_2)$  are two of the four-torsion elements describing the theta-null point on  $A_1$ . If  $n > 3$ , we can use  $f_1(K)[8]$  to describe the 8-torsion above  $\langle f_1(T''_1), f_1(T''_2) \rangle$  and iterate the process.

Once we reach the second last step  $f_{n-1} : A_{n-2} \rightarrow A_{n-1}$ , we cannot inherit the 8-torsion above the  $K_2$  part of the  $A_{n-2}$  anymore. However, thanks to the assumption that we have to two points of order  $2^{n+2}$  on  $E_1 \times E_2$  above  $K$  forming an isotropic group, we can use the same exact strategy using the images of such points. We explain how to relax this condition in Sect. 4.2.

In the last step  $f_n : A_{n-1} \rightarrow E'_1 \times E'_2$ , we map onto an elliptic product. Even though we can reuse the same computational strategies when we stay in the theta model, for most of the cryptographic applications we have to explicitly recover the equations of the curves  $E'_1$  and  $E'_2$ , and we also have to have map points onto these curves. We describe how to do so in the Section “Splitting Isogeny”.

**Gluing Isogeny.** In this section, we focus on the first step of the isogeny chain, an isogeny originating from an elliptic product; let  $f : E_1 \times E_2 \rightarrow A$  be such an isogeny. Theta structures on elliptic products  $E_1 \times E_2$  satisfy some additional properties with respect to level-two theta coordinates. We refer to Dupont’s PhD thesis [16] for background material. For the case at hand, we briefly recall some fundamental facts.

Theta coordinates of level  $(2, 2)$  are indexed by a pair of elements in  $K(2)$ . A level-two theta coordinate  $U_{i,j}$  is said to be *even* if  $i \cdot j^T = 0 \pmod{2}$ ; otherwise it is said to be *odd*. Moreover, at most one of the even indices  $(i, j)$  satisfies  $U_{i,j}(0) = 0$ , and there is exactly one zero even index if and only if the theta structure is associated with a product of two elliptic curves.

Given level-two theta coordinates  $(\theta_i(P))_i$ , we can compute the square of its level-(2, 2) theta coordinates as

$$U_{i,j}^2(P) = \sum_t (-1)^{i \cdot t^T} \theta_t(P) \theta_{t+j}(P).$$

In [16, Proposition 6.5], Dupont shows that a theta-null point  $(\theta_i(0))_i$  comes from the product theta structure of two elliptic curves if and only if  $U_{11,11}(0) = 0$ . This means that if we are working on a product structure, all the coordinates of the dual theta-null point  $(\alpha : \beta : \gamma : \delta)$  are non-zero since  $(U_{00,00}(0) : U_{10,00}(0) : U_{01,00}(0) : U_{11,00}(0)) = (\alpha : \beta : \gamma : \delta)$ . However, when we perform a change of basis, we might move the zero even index of the level-(2, 2) theta-null point around, and potentially we might have one of the dual theta-null coordinates equal to zero.

In fact, unless  $A$  is a product of elliptic curves (which would be the case if the kernel is a product kernel), then we know that one of  $\alpha, \beta, \gamma, \delta$  is zero. If it were not the case, we could compute  $f(P)$  from  $P$ . But since we work with theta coordinates of level two, on the product  $E_1 \times E_2$  we are really working with the product of Kummer lines  $E_1/\pm 1 \times E_2/\pm 1$ . The automorphism group by which we quotient is thus  $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$  compared to  $\mathbb{Z}/2\mathbb{Z}$  when working on the Kummer surface  $A/\pm 1$  of an abelian surface with a non product principal polarisation. Thus, when going from  $P$  to  $f(P)$ , there is an ambiguity coming from an action of  $\mathbb{Z}/2\mathbb{Z}$ , which can only be resolved by either taking a square root, or as we will explain next, by using extra information coming from the arithmetic of  $E_1 \times E_2$  (rather than  $E_1/\pm 1 \times E_2/\pm 1$ ).

Let us handle the case where one of the coordinates of the dual theta-null point  $(\alpha : \beta : \gamma : \delta)$  is zero; let us first analyse the case  $\alpha = 0$ . In Algorithm 5, we normalised everything with respect to  $\delta$ . This actually simplifies the codomain computation. We explain how to do so in Algorithm 8.

---

**Algorithm 8.** Special Codomain,  $\alpha = 0$

---

**Input:** Theta coordinates of  $T_1''$  and  $T_2''$ , where  $T_i''$  is a 8-torsion point lying above the  $K_2$  part of the symplectic four-torsion basis inducing the theta-null point.

**Output:** Dual theta-null point  $(0 : \beta : \gamma : 1)$ , the “inverse” of the dual theta-null point

$(0 : \beta^{-1} : \gamma^{-1} : 1)$  and the theta-null point  $(a' : b' : c' : d')$  on  $A$ . (▷) Case  $\alpha = 0$

1:  $(0, x\beta, y\gamma, y\delta) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_1''}, y_{T_1''}, z_{T_1''}, w_{T_1''})$  (▷) Cost: 4S

2:  $(0, w\beta, z\gamma, w\delta) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{T_2''}, y_{T_2''}, z_{T_2''}, w_{T_2''})$  (▷) Cost: 4S

3: Compute the inverse of  $(y\gamma, w\beta, y\delta, w\delta)$  using batched inversions. (▷) Cost: 9M + 1I

4:  $\beta \leftarrow w\beta \cdot (w\delta)^{-1}$

5:  $\gamma \leftarrow y\gamma \cdot (y\delta)^{-1}$

6:  $\beta^{-1} \leftarrow w\delta \cdot (w\beta)^{-1}$

7:  $\gamma^{-1} \leftarrow y\delta \cdot (y\gamma)^{-1}$

8:  $(a', b', c', d') \leftarrow \mathcal{H}(0, \beta, \gamma, 1)$

9: **return**  $(0, \beta, \gamma, 1), (0, \beta^{-1}, \gamma^{-1}, 1), (a', b', c', d')$  (▷) Total cost: 8S + 13M + 1I

---



As explained above, mapping points under this isogeny requires extra care. If we simply use Algorithm 6, we cannot retrieve the first coordinate of a point. To be precise, if we want to evaluate  $f$  at the point  $(\theta_i^{E_1 \times E_2}(P))_i$ , we have

$$\mathcal{H} \circ \mathcal{S}((\theta_i^{E_1 \times E_2}(P))_i) = (0, \beta \tilde{\theta}_{10}^A(f(P)), \gamma \tilde{\theta}_{01}^A(f(P)), \delta \tilde{\theta}_{11}^A(f(P))). \tag{4}$$

Multiplying by  $\beta^{-1}, \gamma^{-1}$  and  $\delta^{-1}$  the components  $\beta \tilde{\theta}_{10}^A(f(P)), \gamma \tilde{\theta}_{01}^A(f(P)), \delta \tilde{\theta}_{11}^A(f(P))$ , we retrieve all the dual components but  $\tilde{\theta}_{00}^A(f(P))$ .

The component  $\tilde{\theta}_{00}^A(f(P))$  can be computed using the additional information coming from the theta structure. Let  $T'_1$  be the point above  $T_1 \in K_2$  as in the previous section. Then,

$$\mathcal{H} \circ \mathcal{S}((\theta_i^{E_1 \times E_2}(P + T'_1))_i) = (0, \beta \tilde{\theta}_{00}^A(f(P)), \gamma \tilde{\theta}_{11}^A(f(P)), \delta \tilde{\theta}_{01}^A(f(P))). \tag{5}$$

However, multiplying the component  $\beta \tilde{\theta}_{00}^A(f(P))$  by  $\beta^{-1}$  is not enough since we are working up to projective factors.

Once we recover  $\tilde{\theta}_{10}^A(f(P)), \tilde{\theta}_{01}^A(f(P)), \tilde{\theta}_{11}^A(f(P))$  from Eq. 4, we can compute  $\lambda \tilde{\theta}_{01}^A(f(P))$  from Eq. 5 for some projective factor  $\lambda$ : we simply multiply the last component of  $\mathcal{H} \circ \mathcal{S}((\theta_i^{E_1 \times E_2}(P + T'_1))_i)$  by  $\delta^{-1}$ . If  $\tilde{\theta}_{01}^A(f(P)) \neq 0$ , we can actually compute the inverse of the projective factor by  $\tilde{\theta}_{01}^A(f(P)) / (\lambda \tilde{\theta}_{01}^A(f(P)))$ . Otherwise, we repeat the same process with the second last component of  $\mathcal{H} \circ \mathcal{S}((\theta_i^{E_1 \times E_2}(P + T'_1))_i)$ .

**Algorithm 9.** Special Evaluation,  $\alpha = 0$

**Input:** Theta coordinates of  $P$  and  $P + T'_1$  and the “inverse” of the dual theta-null point  $(0 : \beta^{-1} : \gamma^{-1} : 1)$  on  $A$ .

**Output:** Theta coordinates of  $f(P)$ . (▷) Case  $\alpha = 0$

- 1:  $(0, Y_P, Z_P, W_P) \leftarrow \mathcal{H} \circ \mathcal{S}(x_P, y_P, z_P, w_P)$  (▷) Cost: 4S
- 2:  $(0, Y_{P+T_1}, Z_{P+T_1}, W_{P+T_1}) \leftarrow \mathcal{H} \circ \mathcal{S}(x_{P+T_1} : y_{P+T_1} : z_{P+T_1} : w_{P+T_1})$  (▷) Cost: 4S
- 3:  $(Y'_{f(P)}, Z'_{f(P)}, W'_{f(P)}) \leftarrow (\beta^{-1} \cdot Y_P, \gamma^{-1} \cdot Z_P, W_P)$
- 4: **if**  $Z'_{f(P)} \neq 0$  **then**
- 5:  $\lambda^{-1} \leftarrow Z'_{f(P)} / W_{P+T_1}$
- 6: **else**
- 7:  $Z'_{f(P+T_1)} \leftarrow \gamma^{-1} \cdot Z_{P+T_1}$
- 8:  $\lambda^{-1} \leftarrow W'_{f(P)} / Z'_{P+T_1}$
- 9:  $X'_{f(P)} \leftarrow \lambda^{-1} \cdot \beta^{-1} \cdot Y_{P+T_1}$
- 10:  $(x_{f(P)}, y_{f(P)}, z_{f(P)}, w_{f(P)}) \leftarrow \mathcal{H}(X'_{f(P)}, Y'_{f(P)}, Z'_{f(P)}, W'_{f(P)})$
- 11: **return**  $(x_{f(P)}, y_{f(P)}, z_{f(P)}, w_{f(P)})$  (▷) Total cost: 8S + 5M + 1I

Once we have  $\lambda$ , we extract  $\tilde{\theta}_{00}^A(f(P))$  from the second component of  $\mathcal{H} \circ \mathcal{S}((\theta_i^{E_1 \times E_2}(P + T'_1))_i)$ : we multiply the second component of  $\mathcal{H} \circ \mathcal{S}((\theta_i^{E_1 \times E_2}(P + T'_1))_i)$  by  $\lambda^{-1} \cdot \beta^{-1}$ . Finally, we obtain the image of the point  $P$  under  $f$  via

$$\mathcal{H}(\tilde{\theta}_{00}^A(f(P)), \tilde{\theta}_{10}^A(f(P)), \tilde{\theta}_{01}^A(f(P)), \tilde{\theta}_{11}^A(f(P))).$$

We summarise everything in Algorithm 9. Note that for both Algorithm 8 and 9, the case for  $\beta, \gamma$  or  $\delta = 0$  follows almost identically, see the implementation for a concrete example of how all cases can be considered concisely. We note that Algorithm 9 requires the knowledge not only of the theta coordinates of  $P$ , but also of  $P + T'_1$ . From the knowledge of the  $(\theta_i(P))$  and  $(\theta_i(T'_1))$ , we may only recover  $(\theta_i(P \pm T'_1))$ , hence extract  $(\theta_i(P + T'_1))$  via a square root, consistent with the fact that in a gluing isogeny we have an ambiguity for images coming from an action by  $\mathbb{Z}/2\mathbb{Z}$ . Luckily we can compute this addition on each elliptic curve separately, using Weierstrass coordinates, before switching to the level-two theta coordinates on the surface  $E_1 \times E_2$ .

**Splitting Isogeny.** In this last step of the isogeny chain, we need to compute an isogeny  $f : A \rightarrow E'_1 \times E'_2$  mapping onto an elliptic product. We can compute the theta-null point of  $E'_1 \times E'_2$  and mapping points under  $f$  using Algorithms 5 and 6. However, we still need to retrieve the explicit equations for the curves  $E'_1$  and  $E'_2$ . This can be done using level-(2, 2) theta coordinates  $U_{i,j}$ .

Since the theta structure on the image surface underlies an elliptic product, we know that one of the even indices – say  $(i, j)$  – of the level-(2, 2) theta-null point is equal to zero. Also, we know that if we compute a symplectic automorphism  $\psi$  mapping  $(i, j)$  onto  $(11, 11)$ , the action of  $\psi$  on the theta-null point obtained via Algorithm 5 gives back a theta-null point associated with the product theta structure.

From the above, it can be seen that there are ten distinct even indices. For each of these indices, we computed a symplectic automorphism sending this index to  $(11, 11)$ . For efficiency reasons, we hard-coded the action of each of the symplectic automorphisms onto theta coordinates of level two in the reference implementation. These symplectic automorphisms and their actions have been derived from [39, p. 28] using the following sequential steps.

Let  $(i, j)$  be the even index such that  $U_{i,j}(0) = 0$ , and, for ease of notation, let  $(a_{00} : a_{10} : a_{01} : a_{11})$  be the underlying theta-null point.

1. If  $i = j = 00$ , we act by the symplectic automorphism with matrix form

$$\begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

We then obtain the theta-null point  $(a_{00} : \sqrt{-1} \cdot a_{10} : a_{01} : \sqrt{-1} \cdot a_{11})$ , which means that  $U_{10,00}(0) = 0$ .

2. If  $j = 00$  and  $i \neq 00$ , we act by  $\mathcal{H}$ , which swaps the roles of  $i$  and  $j$ . We can now assume that  $j \neq 00$ .
3. Let  $A$  be any invertible matrix such that  $A \cdot j^T = 11^T$ . Then, the action of the symplectic automorphism with matrix

$$\begin{pmatrix} A & 0 \\ 0 & A^{T^{-1}} \end{pmatrix}$$

maps the theta-null point  $(a_{00} : a_{10} : a_{01} : a_{11})$  to  $(a_{00} : a_{10 \cdot A^T} : a_{01 \cdot A^T} : a_{11 \cdot A^T})$ . This means that  $U_{i',j'}(0) = 0$ , where  $i' = i \cdot A$  and  $j' = 11$ . We can now assume that  $j = 11$ .

4. Now, either  $i = 00$  or  $i = 11$ . If  $i = 11$ , we are done. Otherwise, we act by the symplectic automorphism with matrix form

$$\begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

We then obtain the theta-null point  $(a_{00} : \sqrt{-1} \cdot a_{10} : \sqrt{-1} \cdot a_{01} : a_{11})$ , which means that  $U_{11,11}(0) = 0$ .

Thus, we can assume we are now working on the product theta structure.

If  $(a : b : c : d)$  is a theta-null point on  $E_1 \times E_2$ , from Proposition 3, it follows that  $(a : b)$  is a theta-null point for  $E_1$  and  $(b : d)$  is a theta-null point for  $E_2$ . Also, if  $f(P) = (P_1, P_2) \in E_1 \times E_2$  is represented in theta coordinates as  $(x : y : z : w)$ , we have that  $(x : y)$  is the representation of  $P_1$  in theta coordinates for  $E_1$  and  $(y : w)$  is the representation of  $P_2$  in theta coordinates for  $E_2$ . Finally, to convert from theta coordinates to the Montgomery model we can use the formulae in [43, Ch. 7, Appendix A.1], also rederived in Example 4.

## 4.2 Computing Isogenies Without Extra Isotropic Information

In this section, we relax the condition on the two points of order  $2^{n+2}$  on  $E_1 \times E_2$  above  $K$  forming an isotropic group. This does not represent a problem when computing a  $(2^n, 2^n)$ -isogeny, except for the two last steps. We discuss two cases: when we can work with  $2^{n+2}$ -torsion, and when we cannot.<sup>7</sup>

Let us discuss the former case. Let  $K = \langle P_1, P_2 \rangle \subset E_1 \times E_2$ . To apply the previous algorithm, we would like to have an isotropic  $\langle P_1'', P_2'' \rangle$  above  $K$  such that  $P_i = [4]P_i''$ . However, it suffices to pick any  $Q_1'', Q_2''$ , not necessarily isotropic, as long as  $P_i = [4]Q_i''$ . Indeed, one can check that applying the algorithm of Sect. 4.1 on these  $Q_i''$  gives a theta-null point that differs from the one given by isotropic  $P_i''$  by an automorphism of the theta group (see Sect. 2.1) induced by a symplectic automorphism. Hence, it still corresponds to the correct codomain, but with a different theta structure. We refer to [43, Ch. 7, Example B.4] for more details.

In the latter case, we cannot use the  $2^{n+2}$ -torsion at all. A way to circumvent this problem is to use square roots to compute the codomains for the last two steps. Once we have the codomain, the image evaluation is unaffected. There is no way to avoid the square root computations: the theta-null point requires a theta structure of level two, so in particular a full basis of the two-torsion and some extra information on the four-torsion. If we do not have the  $2^{n+2}$ -torsion

<sup>7</sup> For instance, it is preferable not to work with the  $2^{n+2}$ -torsion when it is defined over a field extension of the base field  $k$ .

at the beginning, we miss the necessary information on the four-torsion at the penultimate step and on the two-torsion on the last step. To reconstruct this information requires making choices, hence taking square roots.

At the penultimate step  $f : A \rightarrow B$ , we have  $T'_1$  and  $T'_2$  of four-torsion but not the 8-torsion points  $T''_1$  and  $T''_2$  anymore. This means that on the codomain, we only have the two-torsion determined. We have several choices of possible compatible theta structure, but we still want to use the information at hand.

Let  $(\alpha : \beta : \gamma : \delta)$  be the dual theta-null point on  $B$ . Applying Eq. 2 to the theta-null point  $(a : b : c : d)$ , we have

$$\mathcal{H} \circ \mathcal{S}(a : b : c : d) = (\alpha^2 : \beta^2 : \gamma^2 : \delta^2). \tag{6}$$

Also, since  $f(T'_1)$  is in  $K_1$  for the dual theta structure, we have

$$\left( \tilde{\theta}_i^B(f(T'_1)) \right)_i = (\beta : \alpha : \delta : \gamma).$$

Therefore, from Eq. 2,

$$\mathcal{H} \circ \mathcal{S}((\theta_i^A(T'_1))_i) = (\alpha\beta, \alpha\beta, \gamma\delta, \gamma\delta). \tag{7}$$

Fix  $\alpha = 1$ . From Eq. 6, we can compute any square root of  $\beta^2$  for  $\beta$  and any square root of  $\gamma^2$  for  $\gamma$ . From Eq. 7 and  $\beta$  we can recover the correct lifting of  $\gamma\delta$ , and in turn, we can recover  $\delta$ . The four choices we can make on the square roots of  $\gamma^2$  and  $\delta^2$  describe different theta structures underlying the same abelian surface since they differ by the action of a symplectic automorphism [43, Ch. 7, Example B.3].

At the last step, we only have  $T_1$  and  $T_2$ . As a result, we can only recover the squares  $(\alpha^2 : \beta^2 : \gamma^2 : \delta^2)$  of the dual theta-null point  $(\alpha : \beta : \gamma : \delta)$ . We can fix  $\alpha = 1$  and compute  $\beta, \gamma, \delta$  via three square roots. Once again, we can check that these 8 choices all come from a valid theta structure [43, Ch. 7, Example B.3].

To sum up, if the  $2^{n+2}$ -torsion is available, we need no square root. If the  $2^{n+1}$ -torsion is available, we need two square roots. If only the  $2^n$ -torsion is available, we need  $2 + 3 = 5$  square roots.

## 5 Implementation

We have implemented the computation of an isogeny between elliptic products in the theta model using both the programming language Rust and the computer algebra system SageMath version 10.2. The SageMath implementation has been designed to follow the API of isogenies between elliptic curves and is intended to be a tool in both experimentation and in constructing proof-of-concept implementations of isogeny-based cryptographic primitives. For those who have previously relied on the SageMath implementation of [34], the function `EllipticProductIsogeny(kernel, n)` has been designed to be a drop-in

replacement for the  $(2^n, 2^n)$ -isogeny computed using the Richelot correspondence and the algorithms presented in [45].

The Rust implementation has been designed with constructive cryptographic implementations in mind, and in particular, it has been written to be constant time.<sup>8</sup> The finite field arithmetic and certain elliptic curve functions have been adapted from the `crr1` library [35] maintained by Thomas Pornin as well as other ongoing collaborations. An effort has been made to ensure the code is (reasonably) flexible so that without too much tweaking, this work can be ported to other Rust projects. As an example of this flexibility, we show timings of isogenies of various lengths between elliptic products over three distinct base fields.

Both the SageMath and Rust implementations are made available via the following GitHub repository: <https://github.com/ThetaIsogenies/two-isogenies>.

### 5.1 Performance

In this section, we include the performance of our algorithm for three distinct isogeny chains between elliptic products over a range of base fields. We include the timings for both the constant-time Rust implementation as well as the proof-of-concept SageMath implementation, together with a comparison to previous work on isogenies between elliptic products in the Mumford model [34] using the optimisations introduced in the implementation of [2].

This triplet of comparisons has a twofold advantage. Firstly, the Rust implementation we present is the first (to our knowledge) constant time implementation of dimension two isogenies between elliptic products. By including the timings of both our Rust implementation and the SageMath implementation, we hope that researchers can estimate a performance gain if they were to write efficient and cryptographically minded implementations following the proof-of-

**Table 2.** Running times of computing the codomain and evaluating a  $(2^n, 2^n)$ -isogeny between elliptic products over the base field  $\mathbb{F}_{p^2}$ . Times were recorded on an Intel Core i7-9750H CPU with a clock-speed of 2.6 GHz with turbo-boost disabled.

$\log p$	$n$	Codomain			Evaluation		
		Theta Rust	Theta SageMath	Richelot SageMath [34]	Theta Rust	Theta SageMath	Richelot SageMath [34]
254	126	<b>2.13 ms</b>	108 ms	1028 ms	<b>161 <math>\mu</math>s</b>	5.43 ms	114 ms
381	208	<b>9.05 ms</b>	201 ms	1998 ms	<b>411 <math>\mu</math>s</b>	8.68 ms	208 ms
1293	632	<b>463 ms</b>	1225 ms	12840 ms	<b>17.8 ms</b>	40.8 ms	1203 ms

<sup>8</sup> The implementation assumes the kernel generators are *good* with respect to them generating an isogeny between elliptic products. Designing the algorithm to run in constant time with malformed input extends beyond the goals of this paper but may be necessary for protection against side-channel attacks against schemes which rely on this algorithm.

concept scripts which currently exist in the higher-dimensional isogeny-based cryptography literature.

Secondly, our SageMath implementation allows an honest comparison of the isogenies in the theta model to the Richelot isogenies in the Mumford model. We compare against the implementation of [34] together with the additional optimisations introduced for the proof-of-concept of [2] which offered more than a two times speed up by optimising both the arithmetic on Jacobians as well as the isogenies themselves.

The run-times displayed in Table 2 were captured on an Intel Core i7-9750H CPU with a clock-speed of 2.6 GHz with turbo-boost disabled for stable measurements. The Rust code was compiled with the Rust compiler version 1.80.0-nightly with the flag `-C target-cpu=native` to allow the compiler to use CPU specific opcodes (specifically, `mulx` for the finite field arithmetic). The arithmetic is written using Rust, rather than optimised assembly for each base field; the inclusion of which would allow dramatically faster results, especially for base fields with large characteristic. This form of optimisation is better suited to particular protocols, and we would expect to see this in optimised implementations of isogeny-based cryptographic primitives.

Comparing our SageMath implementation (version 10.2) to the isogeny chain in the Mumford model, we find that the codomain computation is consistently faster by a factor of ten, while the image computation is more than twenty times faster. For the smaller characteristics studied, the Rust implementation is approximately forty times faster than the same algorithm written in SageMath, but this gap closes significantly for larger primes. For example, the FESTA sized parameters run only 2.5 times faster than the SageMath code. Note that the Rust implementation has been written to run in constant time and so the underlying arithmetic between these two implementations is incomparable.<sup>9</sup>

We note here that an alternative and faster implementation of  $(2, 2)$ -isogenies in the Mumford model is available in [19]. In this work, Kunzweiler uses Jacobians of hyperelliptic curves in specific models which allows  $(2, 2)$ -isogeny chains to be computed particularly efficiently. In the initial treatment of this work, isogenies between elliptic products were not considered, leading to FESTA [2] and other projects to rely on [34]. However, Kunzweiler's work can be adapted to the case of isogenies between elliptic products. Additionally, Kunzweiler also has an unpublished SageMath implementation of  $(2, 2)$ -isogenies using Kummer surfaces in the Mumford model rather than in the Jacobian model that she kindly provided us.<sup>10</sup> Comparing our results to the computations of Kummer surfaces in the Mumford model is a fairer comparison as we work with level-two theta

---

<sup>9</sup> It is not surprising to see this gap close though, as we expect for very large characteristic that the SageMath overhead becomes negligible compared to the cost of the arithmetic. As such, the comparisons of the two run-times boil down to comparing the Rust finite field arithmetic against the SageMath calls to the optimised arithmetic of the C libraries it is built upon.

<sup>10</sup> Kunzweiler's isogenies between elliptic products using both Jacobians and Kummer surfaces are now available via GitHub [20].

**Table 3.** Comparison of the SageMath running times for a  $(2^n, 2^n)$ -isogeny between elliptic products in the theta model against Kunzweiler’s implementation in the Mumford model using both Jacobians and Kummer surfaces [20]. Times were recorded on an Intel Core i7-9750H CPU with a clock-speed of 2.6 GHz with turbo-boost disabled.

$\log p$	$n$	Codomain			Evaluation		
		Theta	Jacobian	Kummer	Theta	Jacobian	Kummer
254	126	108 ms	760 ms	467 ms	5.43 ms	66.7 ms	18.4 ms
381	208	201 ms	1478 ms	858 ms	8.68 ms	119 ms	31.4 ms
1293	632	1225 ms	9196 ms	5150 ms	40.8 ms	593 ms	170 ms

coordinates, which are also on Kummer surfaces. Comparing against this implementation, the codomain computation in the theta model is around four times faster than in the Mumford model, and evaluations are around four times faster. We give detailed comparison timings in Table 3.

Although theta coordinates are faster, working in the Mumford model is interesting when the level-two theta coordinates are not rational, which would require using theta coordinates on a field extension. Since our domain is a product of elliptic curves, the theta coordinates are rational when each elliptic curve is described by rational theta coordinates. Following the discussion in Sect. 2.3, an elliptic curve  $E$  has rational theta coordinates when  $E[4]$  is rational.

**Comparison with Dimension One.** In Table 4, we provide a timing comparison using SageMath between a  $2^n$ -isogeny in dimension one using the efficient formulae of [36] to our formulae in dimension two over the same base field. The dimension two isogeny has degree  $2^{2n}$  so is expected to be slower. Our timings show a consistent factor-two slow down both for the codomain and image computations in dimension two compared to dimension one. This is essentially the best we could hope given the degrees, and actually better than expected.

The dominating costs of a  $2^n$ -isogeny are the intermediate doublings and images. In the following we consider the more costly doublings and images in dimension two which arise from avoiding inversion when computing the codomain. First of all, we have around twice as many doublings and images in dimension two than in dimension one because the kernel is of rank two. The cost of doubling in dimension one is  $4\mathbf{M} + 2\mathbf{S}$  compared to  $8\mathbf{M} + 8\mathbf{S}$  in dimension two, and an image is  $4\mathbf{M}$  compared to  $4\mathbf{M} + 4\mathbf{S}$  in dimension two. Thus, while images are twice slower, doublings are around  $2.5\times$  slower, and the intermediate codomain computations are also slower. Furthermore, a lot of doublings are done on the first step of the chain to get the first kernel, so on the elliptic product.

While it might seem at first glance that these doublings would only incur a twofold slowdown, in practice, for the gluing images, we need to compute these points in affine  $(x, y)$  coordinates rather than  $x$ -only coordinates to allow access to addition laws.<sup>11</sup>

<sup>11</sup> We could also use differential additions to compute  $[m]P, [m]P + T'_1$ , but this would be more expensive than just doubling in the affine model.

**Table 4.** Comparison of the running times for a  $2^n$ -isogeny in dimension one and dimension two over the same base field. Times were recorded on an Intel Core i7-9750H CPU with a clock-speed of 2.6 GHz with turbo-boost disabled.

$\log p$	$n$	Codomain		Evaluation	
		Montgomery	Theta	Montgomery	Theta
254	126	63 ms	108 ms	2.24 ms	5.43 ms
381	208	136 ms	201 ms	4.4 ms	8.68 ms
1293	632	727 ms	1225 ms	20 ms	40.8 ms

So, all in all, we should expect a slowdown around  $4\times$  for perfect implementations. Our benchmarks show a slowdown slightly less than  $2\times$ , making two-dimensional isogenies perform better than expected (by contrast, the  $2\times$  slowdown for images is consistent with the theory). This is probably due to SageMath overhead and the fact that the dimension one implementation has been designed to allow arbitrary degree rather than only chains of two isogenies and is missing some optimisations. A final caveat is that in dimension one, it is faster to split the  $2^n$ -isogeny using the fast four-isogenies from [11] rather than using two-isogenies – we did not do that in our comparison because we do not have efficient four-isogeny formulae in dimension two yet. Still, taking into account the degrees of the respective isogenies, this shows that our dimension two formulae are quite competitive with the best dimension one formulae.

### 5.2 Implementation Details

In this section, we explain two optimisations we applied in the implementation. The first one is a direct consequence of Remark 8, where we describe how to lower the complexity of the codomain computation by reusing some constants. The second optimisation consists in the application of *optimal strategies* [14] to our case.

**Reduce, Reuse, Recycle.** A simple and obvious optimisation is to reuse as many computations as possible throughout the isogeny chain. As mentioned in Remark 8, for each step on the isogeny chain, we precompute six field elements for doubling with a normalised null point at a cost of  $4\mathbf{S} + 21\mathbf{M} + 1\mathbf{I}$  or eight field elements at a cost of  $6\mathbf{S} + 16\mathbf{M}$  for the projective null point. Knowledge of these values allows the doubling of any theta point on the corresponding theta structure to have a cost of  $8\mathbf{S} + 6\mathbf{M}$  and  $8\mathbf{S} + 8\mathbf{M}$  respectively, but it also allows the following evaluation precomputation cost to be lowered from  $13\mathbf{M}$  and  $6\mathbf{M}$  to  $4\mathbf{M}$  for both cases.

For the gluing isogeny, the basis change is determined from the kernel and so cannot be precomputed. However, the last step at the end of the isogeny chain requires to find a symplectic transformation that maps the zero even index to the position  $(11, 11)$ . As there are only ten even indices, we can precompute ten



symplectic transforms which map any given zero even index to  $(11, 11)$ . Computing the basis change is then only a matter of finding the current zero index and from this, selecting the precomputed matrix and applying the transformation.

**On Inversions.** At each step of the isogeny chain, we compute one inversion for the intermediate codomain. This inversion allows us to reduce the cost of doublings on this codomain from  $8\mathbf{M} + 8\mathbf{S}$  to  $6\mathbf{M} + 8\mathbf{S}$  and the cost of images from  $4\mathbf{M} + 4\mathbf{S}$  to  $3\mathbf{M} + 4\mathbf{S}$ . However, at the end of the isogeny chain, there remain fewer doublings and images to compute, so it would be more efficient to skip this inversion and incur the higher cost. The precise cutoff point would depend on the relative cost of the inversion compared to a multiplication and the number of doublings and evaluations required at each step along the chain. This optimisation has not yet been implemented in our code, where we work with projective null points along the whole chain for simplicity.

**On Square Roots.** As explained in Sect. 4.2, when we do not have the  $2^{n+2}$ -torsion available, we need to compute some square roots at the end of the chain (five square roots in total). This only changes the computation cost of the last two codomains, and do not affect the images computations. The longer the isogeny chain, the less impactful these square roots will be. With our SageMath implementation, we observed that the impact of these five square roots is completely negligible for the chains we consider.

**Optimal Strategies.** As is now standard with computing long isogeny chains, we can reduce the complexity of isogeny chains from a quadratic number of edges in the graph of doublings and evaluations to quasi-linear following the “optimal strategies” introduced in [14]. Essentially, the saving comes from reducing the total number of doublings when computing the kernel for each step in the chain by pushing through intermediate points encountered in the repeated doubling. For isogenies in the theta model, the cost of images is half that of doubling, and so shifting the cost in this way is particularly useful in optimisations.

Although this strategy was first discussed in dimension one for the case of isogenies between elliptic curves, using it in dimension two is a natural generalisation—see for instance [5]. For the dimension one case, the strategy is computed from balancing the costs of doubling and evaluating the kernel generator through the chain. In dimension two, the  $(2, 2)$ -isogeny is generated by a *pair* of elements which means twice the number of evaluations, but as the pair of elements must also be doubled to obtain the kernel for each step, essentially nothing changes. The cost weighting for the optimal strategies is a ratio between doublings and evaluations, which means we can naively use an identical method as described in [14] to compute a strategy for our isogeny chain.

Implementing the strategy with the weighting of doublings and images at a cost of  $(2 : 1)$ , we find an approximate ten times speed up in comparison to an implementation with no strategy. Concretely, for the Rust implementation of the isogeny chain of length  $n = 208$ , we see a speed up from 107 ms to 11.4 ms.

However, unlike the isogeny chains between elliptic curves, the isogeny chain between elliptic products in our implementation does not have the same costs

for every step. For steps in the chain between generic theta structures, the cost weighting is indeed (2 : 1). However, for the first gluing isogeny, doubling an element on the product structure has a cost of 12S + 12M while the cost for the image is much more expensive.

To compute the image of a point  $P \in E_1 \times E_2$  one must first compute the shift  $P + T'_1$  for a cost of 10S + 32M to projectively add a pair of points. Then, for each of these two points on the product, there is a cost of 4M to compute the corresponding theta point from elliptic curve coordinates, and an additional 16M required perform the matrix multiplication for the basis change to ensure a compatible representation. Altogether, this precomputation costs 10S + 72M. Given the theta point corresponding to the pair of points on the product structure, there is still then the final cost of 8S + 5M + 1I for the special image itself. Furthermore, for this to be implemented in constant time, both branches depending on whether a coordinate is zero or not must be evaluated, raising the practical cost to 8S + 10M + 1I.

On the whole, a gluing image costs 18S + 82M + 1I, making it approximately seven times the cost of the doubling for this first step and fourteen times the cost of a regular image. Visualising the graph of doublings and images as in [14, Figure 2], this means we must weigh the cost of moving down the left most branch with the product doubling and the first step right from the leftmost branch with this high-cost gluing image.

Taking this into account, an optimised strategy for the isogeny between elliptic products for our formula must be tweaked from the original case to find the right balance of doublings and expensive images from this left branch. Applying this modification, we are able to find the “proper” optimised strategy, which further reduces the run-time of the isogeny chain computation by approximately 2%.<sup>12</sup> For the same chain as above, we see a computation time improve from 11.4ms to 11.2ms. For an explicit description for computing the optimised strategy with a different costs on the left-most branch, see the implementation.

### 5.3 An Application: FESTA

As an explicit, cryptographic example of the new isogeny formula, we can take our implementation and use it to compute the isogeny between elliptic products which is required within the decryption algorithm of the isogeny-based public key encryption protocol FESTA-128 [2]. Concretely, this requires computing an isogeny of length  $n = 632$ , where the base field has a characteristic with  $\log p = 1293$  bits, and the evaluation of a pair of points on the elliptic product  $L_1 = (R_1, R_2)$  and  $L_2 = (S_1, S_2)$ ,  $L_i \in E_1 \times E_2$ .

A direct swap from the isogeny chain derived from the Richelot correspondence used in the FESTA proof-of-concept would require using Sect. 4.2 to com-

---

<sup>12</sup> As an aside, in the original discussion of the optimised costings, it is shown that a 2–3% improvement is gained by moving from a balanced to optimised strategy. Seeing a similar saving from the naive (2 : 1) weighted optimisation to one carefully handling the cost of the gluing step is then within our expectations.

pute the final two steps without the eight-torsion above the kernel. An implementation of this is available in SageMath, but for the purpose of FESTA, we instead propose to tweak the 128-bit parameter set to instead allow for the additional torsion information to be known, allowing the isogeny chain to be computed as fast as possible while only including an additional two bits in the masked torsion data.<sup>13</sup>

We find that our SageMath implementation of the codomain computation has a ten times speed up compared to the proof-of-concept code accompanying [2], and evaluating the pair of points is now thirty times faster. As a hint to what approximate running times may be for FESTA, computing the codomain and both images using our Rust implementation takes only 563ms, a 2.5 times speed up over the SageMath implementation. Note that these computation are precisely that of the final row of Table 2. Optimisations of the finite field arithmetic could offer substantial speed ups, as seen in the optimised assembly implementations for large characteristic SIDH [18, Table 2.1] and the efficient algorithms of [22].

In SageMath, the novel algorithms we present here offer a four times speed up in decryption, with run-times for FESTA-128 being reduced from 20.7s to only 5.4s. When computing the dimension two isogeny in the theta model, the time spent for the  $(2^n, 2^n)$ -isogeny shrinks from 70% of the run-time to only 25%, with the remaining computation time spent in dimension one, computing various discrete logarithms and Weil pairings to complete the decryption routine.

## 6 Conclusions

In this paper, we have described and implemented formulae to compute  $(2^n, 2^n)$ -isogenies between elliptic products in the theta model. The main goal was to provide a comprehensive and self-contained treatment of the theta model, specialising to the two-dimensional case.

Our algorithm significantly outperforms the previous method in [2, 34]: in SageMath, the codomain computation is ten times faster, while the isogeny evaluation is more than twenty times faster. The implementation in Rust has been written to run in constant time, with cryptographic implementations in mind. It runs up to forty times faster than the same algorithm written in SageMath.

We tested our algorithm on the proof-of-concept implementation in [2] and showed a fourfold speed up in decryption, highlighting that the slowest part is now given by the computations in dimension one. Furthermore, our SageMath implementation has been designed to allow protocols whose implementation relies on the previous proof-of-concept in [2] to be easily upgradeable, allowing the theta model code to be used in many more projects without too much work. Ultimately, the aim is to provide a new tool to facilitate research in higher-dimensional isogeny-based cryptography, allowing us to better understand the practical role of higher-dimensional isogenies in constructive applications.

<sup>13</sup> SageMath benchmark of FESTA isogeny.

**Acknowledgments..** Huge thanks are given to Thomas Pornin for his advice and previous collaborations, both of which were instrumental in the design and implementation of the constant time Rust implementation. We also thank Sabrina Kunzweiler for fruitful discussion and the anonymous reviewer for pointing out additional applications of the theta model outside isogeny-based cryptography.

## A The General Case

In this section, we briefly explain how to compute a general  $(2^n, 2^n)$ -isogeny between Kummer surfaces (with decomposable or indecomposable polarisations).

In theory, using the theta model would give an uniform approach to handle both Jacobian of hyperelliptic curves of genus two and product of elliptic curves. However, in order to achieve the best performance, we rely on a theta model of level two rather than higher level  $n > 2$  (since in level  $n$  we have  $n^g$  theta coordinates), which yields the following technical difficulty.

Let  $A$  be a principally polarised abelian surface. If  $A$  corresponds to a Jacobian, then the level-two theta coordinates give an embedding of the Kummer surface  $A/\pm 1$ . However, as explained in the main text, if  $A = E_1 \times E_2$  is a product of two elliptic curves (with their product polarisation), then the level-two theta coordinates give an embedding of the product of Kummer lines  $(E_1/\pm 1) \times (E_2/\pm 1)$ . In particular, we do not get an embedding of  $(E_1 \times E_2)/\pm 1$  but of a further quotient.

As a consequence, for a gluing image  $(E_1/\pm 1) \times (E_2/\pm 1) \rightarrow A/\pm 1$ , knowing a point  $P = (\pm P_1, \pm P_2)$  is not enough to determine its image in  $A$ : we need extra data. This is why we had to use a special algorithm for the gluing isogeny in Sect. 4. In practice, the gluing case can be detected when some of our intermediate theta constants are zero. As mentioned above, if we were working in level  $n > 2$ , we would always have enough non-zero theta constants to compute images in all cases (by [29]), but we need an alternative strategy for  $n = 2$ .

We now explain how to deal with all cases for a  $(2^n, 2^n)$ -isogeny  $A \rightarrow B$  in level two with kernel  $K$ . In the main text, we already dealt with the case where both  $A, B$  are product of elliptic curves, but none of the intermediate abelian surfaces are.

1. **When the codomain  $B$  is not a product.** In this case we proceed as in the main algorithm, except we do not need to find a product theta structure in the end. If needed, to recover  $B$  as a Jacobian and to convert between theta coordinates and Mumford coordinates, we can use Thomae’s formula and the conversion formula from [32], see also [8, 9, 47].
2. **When the domain  $A$  is not a product.** If  $A$  and the kernel are already described by theta coordinates, we first need to do a symplectic change of theta coordinate to make the kernel compatible with our theta structure. To compute this change of basis, we can proceed as in Sect. 2.3 by taking suitable traces under the symmetric elements of the theta group induced by  $K[4]$ . For the case of a product of elliptic curves, we had to give the explicit action of the symmetric theta group element corresponding to a couple of points

of four-torsion of elliptic curves on the product of coordinates. In our case, since we already have theta coordinates, this action is already encoded by our theta structure. We refer to the change of coordinates formulae provided in [12, Theorem 12].

If  $A$ , which is a Jacobian  $\text{Jac}(C)$  under our hypothesis, is described by the curve  $C$  and the kernel  $K$  has its generators given in Mumford coordinates, we first need to convert into theta coordinates, using the formulae of [8, 9, 32, 47] as above. In that case, Kunzweiler has formulae<sup>14</sup> for how to take the fourth-roots in Thomae’s formula that directly give the theta constants compatible with the kernel; this allows to bypass the change of basis step once we have the theta coordinates.

- 3. **When the first step is between elliptic products.** If the chain begins with a  $(2, 2)$ -isogeny between products  $\Phi : E_1 \times E_2 \rightarrow E'_1 \times E'_2$ , the isogeny  $\Phi$  is a diagonal isogeny, i.e.  $\Phi = \begin{pmatrix} \phi_1 & 0 \\ 0 & \phi_2 \end{pmatrix}$ , where  $\phi_i : E_i \rightarrow E'_i$  is a one-dimensional isogeny. This case reduces to first computing the one-dimensional isogenies  $\phi_i$ ’s to encode the first step and then resuming from the resulting elliptic product.

The only other possibility is that we have an isogeny diamond (i.e., a Kani square) with isogenies of degree one (i.e., isomorphisms). Then Kani’s lemma give a  $(2, 2)$ -isogeny  $\Phi$ . For instance if we take  $E_1 = E_2 = E'_1 = E'_2 = E$  and we consider the automorphisms  $\text{Id} : E_i \rightarrow E_i$ ; then we obtain the two-isogeny  $\Phi : (P, Q) \mapsto (P+Q, P-Q)$ , and whose kernel is  $\{(T, T) \mid T \in E[2]\}$ . All other  $(2, 2)$ -isogenies  $E \times E \rightarrow E \times E$  which are not given by diagonal two-isogenies in dimension one are variant of this  $\Phi$  where we apply some automorphisms to  $P$  or  $Q$  before. (This only gives a different kernel when  $j(E) = 0$  or  $j(E) = 1728$  and we have non trivial automorphisms, i.e. different from  $\pm 1$ ).

- 4. **An intermediate abelian surface is a product.** In that case, the easiest solution would be to restart the computation using level  $n = 4$  (which requires 16 coordinates rather than four), or the representation from [23] (which requires eight coordinates), because they give embeddings of the abelian surfaces in both the product and non product case, and allow to treat both cases uniformly.

Another solution is to switch to the representation from [23] on the fly. Let us treat the case of a gluing directly followed by a splitting:  $A \rightarrow E_1 \times E_2 \rightarrow B$ , with  $\Phi_1 : A \rightarrow E_1 \times E_2$  and  $\Phi_2 : E_1 \times E_2 \rightarrow B$ .

The splitting step can be handled as in the main text, where we had to compute a splitting as the last step; namely we can compute a product theta structure on  $E_1 \times E_2$ . The difference is that now, we are not at the last step anymore, so we still need to compute a gluing image afterwards.

For reasons explained above, knowing  $\Phi_1(P)$  in level-two theta coordinates is not enough to compute the gluing  $\Phi_2 \circ \Phi_1(P)$ . As in Sect. 4, for the gluing we need  $\Phi_1(P)$  and  $\Phi_1(P) + T'$  in level-two coordinates, for  $T'$  a point of four-torsion in  $E_1 \times E_2$ .

---

<sup>14</sup> Private communication.

One way to obtain these point is to take  $T'' \in A$  a point of 8-torsion in  $A$ , above  $\ker \Phi_2 \circ \Phi_1$ . We compute a representation of the set  $\{P \pm T''\}$  using the formulae of [23], and we compute  $\Phi_1(P)$ ,  $\{\Phi_1(P \pm T'')\}$  in level-two coordinates. From our choice of  $T''$  we have that  $T' = \Phi_1(T'')$  is a point of four-torsion in  $E_1 \times E_2$ . We do not quite have  $\Phi_1(P)$ ,  $\Phi_1(P) + T'$ , but only  $\Phi_1(P) \pm T'$ . However, from our choice of  $T''$  we have that  $\Phi_2(T')$  is a point of two-torsion, hence the two points  $\Phi_2 \circ \Phi_1(P) \pm \Phi_2(T')$  are the same. This means that we can use our gluing algorithm as before.

The case where we have  $m$  several successive product  $A \rightarrow E_1 \times E_2 \rightarrow E'_1 \times E'_2 \rightarrow \dots \rightarrow B$  can be treated in a similar way, by taking a point  $T''$  of  $2^{m+2}$ -torsion above the kernel of  $A \rightarrow B$ , pushing  $P, P \pm T''$  through the splitting isogeny and the intermediate isogenies, then taking a final gluing isogeny.

## References

1. Basso, A., De Feo, L., Dartois, P., Leroux, A., Maino, L., Pope, G., Robert, D., Wesolowski, B.: SQIsign2D-West: The Fast, the Small, and the Safer. *Cryptology ePrint Archive*, Paper 2024/760 (2024), <https://eprint.iacr.org/2024/760>
2. Basso, A., Maino, L., Pope, G.: FESTA: Fast encryption from supersingular torsion attacks. In: *Advances in Cryptology – ASIACRYPT 2023*. pp. 98–126 (2023). [https://doi.org/10.1007/978-981-99-8739-9\\_4](https://doi.org/10.1007/978-981-99-8739-9_4)
3. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) *Advances in Cryptology – EUROCRYPT 2023, Part V. Lecture Notes in Computer Science*, vol. 14008, pp. 423–447. Springer, Heidelberg, Germany, Lyon, France (Apr 23–27, 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_15](https://doi.org/10.1007/978-3-031-30589-4_15)
4. Chen, M., Leroux, A., Panny, L.: SCALLOP-HD: group action from 2-dimensional isogenies. In: *Public-Key Cryptography - PKC 2024*. pp. 190–216 (2024). [https://doi.org/10.1007/978-3-031-57725-3\\_7](https://doi.org/10.1007/978-3-031-57725-3_7)
5. Chi-Domínguez, J.J., Pizarro-Madariaga, A., Riquelme, E.: Computing Quotient Groups of Smooth Order with Applications to Isogenies over Higher-Dimensional Abelian Varieties. *Cryptology ePrint Archive*, Paper 2023/508 (2023), <https://eprint.iacr.org/2023/508>
6. Chudnovsky, D., Chudnovsky, G.: Sequences of numbers generated by addition in formal groups and new primality and factorization tests. *Advances in Applied Mathematics* **7**(4), 385–434 (1986). [https://doi.org/10.1016/0196-8858\(86\)90023-0](https://doi.org/10.1016/0196-8858(86)90023-0)
7. Cornell, G., Silverman, J.H.: *Arithmetic Geometry*. Springer (1986), <https://doi.org/10.1007/978-1-4613-8655-1>
8. Cosset, R.: *Application des fonctions thêta à la cryptographie sur courbes hyperelliptiques*. Ph.D. thesis (2011)
9. Cosset, R., Robert, D.: Computing  $(\ell, \ell)$ -isogenies in polynomial time on Jacobians of genus 2 curves. *Mathematics of Computation* **84**, 1953–1975 (2015). <https://doi.org/10.1090/S0025-5718-2014-02899-8>

10. Costello, C.: Computing supersingular isogenies on Kummer surfaces. In: Peyrin, T., Galbraith, S. (eds.) *Advances in Cryptology – ASIACRYPT 2018, Part III*. Lecture Notes in Computer Science, vol. 11274, pp. 428–456. Springer, Heidelberg, Germany, Brisbane, Queensland, Australia (Dec 2–6, 2018). [https://doi.org/10.1007/978-3-030-03332-3\\_16](https://doi.org/10.1007/978-3-030-03332-3_16)
11. Costello, C., Hisil, H.: A simple and compact algorithm for SIDH with arbitrary degree isogenies. In: Takagi, T., Peyrin, T. (eds.) *Advances in Cryptology – ASIACRYPT 2017, Part II*. Lecture Notes in Computer Science, vol. 10625, pp. 303–329. Springer, Heidelberg, Germany, Hong Kong, China (Dec 3–7, 2017). [https://doi.org/10.1007/978-3-319-70697-9\\_11](https://doi.org/10.1007/978-3-319-70697-9_11)
12. Dartois, P.: Fast computation of 2-isogenies in dimension 4 and cryptographic applications. Cryptology ePrint Archive, Paper 2024/1180 (2024), <https://eprint.iacr.org/2024/1180>
13. Dartois, P., Leroux, A., Robert, D., Wesolowski, B.: SQIsignHD: New Dimensions in Cryptography. In: Joye, M., Leander, G. (eds.) *Advances in Cryptology – EUROCRYPT 2024*. pp. 3–32. Springer Nature Switzerland, Cham (2024)
14. De Feo, L., Jao, D., Plût, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. Cryptology ePrint Archive, Paper 2011/506 (2011), <https://eprint.iacr.org/2011/506>
15. Decru, T., Maino, L., Sanso, A.: Towards a Quantum-Resistant Weak Verifiable Delay Function. In: Aly, A., Tibouchi, M. (eds.) *Progress in Cryptology - LATINCRYPT 2023*. Lecture Notes in Computer Science, vol. 14168, pp. 149–168. Springer (2023). [https://doi.org/10.1007/978-3-031-44469-2\\_8](https://doi.org/10.1007/978-3-031-44469-2_8)
16. Dupont, R.: Moyenne arithmético-géométrique, suites de Borchardt et applications. Ph.D. thesis, École polytechnique (2006)
17. Gaudry, P.: Fast genus 2 arithmetic based on theta functions. *J. Math. Cryptol.* **1**(3), 243–265 (2007). <https://doi.org/10.1515/JMC.2007.012>
18. Jao, D., Azarderakhsh, R., Campagna, M., Costello, C., De Feo, L., Hess, B., Jalali, A., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Renes, J., Soukharev, V., Urbanik, D.: Supersingular isogeny key encapsulation. Submission to <https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization> (2017), <https://sike.org>
19. Kunzweiler, S.: Efficient computation of  $(2^n, 2^n)$ -isogenies. Cryptology ePrint Archive, Paper 2022/990 (2022), <https://eprint.iacr.org/2022/990>
20. Kunzweiler, S.: Efficient Computation of  $(2^n, 2^n)$ -isogenies (2023), <https://github.com/sabrinakunzweiler/richelet-isogenies>
21. Leroux, A.: Verifiable random function from the Deuring correspondence and higher dimensional isogenies. Cryptology ePrint Archive, Paper 2023/1251 (2023), <https://eprint.iacr.org/2023/1251>
22. Longa, P.: Efficient Algorithms for Large Prime Characteristic Fields and Their Application to Bilinear Pairings. *IACR Transactions on Cryptographic Hardware and Embedded Systems* (3), 445–472 (Jun 2023). <https://doi.org/10.46586/tches.v2023.i3.445-472>
23. Lubicz, D., Robert, D.: Arithmetic on abelian and kummer varieties. *Finite Fields and Their Applications* **39**, 130–158 (5 2016). <https://doi.org/10.1016/j.ffa.2016.01.009>
24. Lubicz, D., Robert, D.: Fast change of level and applications to isogenies. *Research in Number Theory (ANTS XV Conference)* **9**(1) (12 2022). <https://doi.org/10.1007/s40993-022-00407-9>

25. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) *Advances in Cryptology – EUROCRYPT 2023, Part V. Lecture Notes in Computer Science*, vol. 14008, pp. 448–471. Springer, Heidelberg, Germany, Lyon, France (Apr 23–27, 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_16](https://doi.org/10.1007/978-3-031-30589-4_16)
26. Montgomery, P.L.: Speeding the pollard and elliptic curve methods of factorization. *Mathematics of Computation* **48**(177), 243–264 (1987). <https://doi.org/10.1090/s0025-5718-1987-0866113-7>
27. Moriya, T.: IS-CUBE: An isogeny-based compact KEM using a boxed SIDH diagram. *Cryptology ePrint Archive*, Paper 2023/1506 (2023), <https://eprint.iacr.org/2023/1506>
28. Moriya, T.: LIT-SiGamal: An efficient isogeny-based PKE based on a LIT diagram. *Cryptology ePrint Archive*, Paper 2024/521 (2024), <https://eprint.iacr.org/2024/521>
29. Mumford, D.: On the Equations Defining Abelian Varieties. I. *Inventiones Mathematicae* **1** (12 1966). <https://doi.org/10.1007/BF01389737>
30. Mumford, D.: On the Equations Defining Abelian Varieties. II. *Inventiones Mathematicae* **3** (01 1967). <https://doi.org/10.1007/BF01389741>
31. Mumford, D.: On the Equations Defining Abelian Varieties. III. *Inventiones Mathematicae* **3** (01 1967). <https://doi.org/10.1007/BF01425401>
32. Mumford, D.: *Tata Lectures on Theta I*. Birkhäuser, Boston (2007)
33. Nakagawa, K., Onuki, H.: QFESTA: Efficient Algorithms and Parameters for FESTA using Quaternion Algebras. *Cryptology ePrint Archive*, Paper 2023/1468 (2023), <https://eprint.iacr.org/2023/1468>
34. Oudompheng, R., Pope, G.: A Note on Reimplementing the Castryck-Decru Attack and Lessons Learned for SageMath. *Cryptology ePrint Archive*, Paper 2022/1283 (2022), <https://eprint.iacr.org/2022/1283>
35. Pornin, T.: `crr1`: Rust library for cryptographic research, version 0.7.0 (2023), <https://github.com/pornin/crr1>
36. Renes, J.: Computing isogenies between montgomery curves using the action of (0,0). *Cryptology ePrint Archive*, Paper 2017/1198 (2017), <https://eprint.iacr.org/2017/1198>
37. Renes, J., Schwabe, P., Smith, B., Batina, L.:  $\mu$ kummer: Efficient hyperelliptic signatures and key exchange on microcontrollers. In: Gierlichs, B., Poschmann, A.Y. (eds.) *Cryptographic Hardware and Embedded Systems – CHES 2016. Lecture Notes in Computer Science*, vol. 9813, pp. 301–320. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–19, 2016). [https://doi.org/10.1007/978-3-662-53140-2\\_15](https://doi.org/10.1007/978-3-662-53140-2_15)
38. Robert, D.: *Fonctions thêta et applications à la cryptographie*. Ph.D. thesis, Université Henry Poincaré - Nancy 1 (2010)
39. Robert, D.: *Efficient algorithms for abelian varieties and their moduli spaces* (2021), *Habilitation à Diriger des Recherches*
40. Robert, D.: Evaluating isogenies in polylogarithmic time. *Cryptology ePrint Archive*, Report 2022/1068 (2022), <https://eprint.iacr.org/2022/1068>
41. Robert, D.: Some applications of higher dimensional isogenies to elliptic curves (overview of results). *Cryptology ePrint Archive*, Report 2022/1704 (2022), <https://eprint.iacr.org/2022/1704>
42. Robert, D.: Breaking SIDH in polynomial time. In: Hazay, C., Stam, M. (eds.) *Advances in Cryptology – EUROCRYPT 2023, Part V. Lecture Notes in Computer Science*, vol. 14008, pp. 472–503. Springer, Heidelberg, Germany, Lyon, France (Apr 23–27, 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_17](https://doi.org/10.1007/978-3-031-30589-4_17)










43. Robert, D.: Some notes on algorithms for abelian varieties. Cryptology ePrint Archive, Paper 2024/406 (2024), <https://eprint.iacr.org/2024/406>
44. Robert, D., Sarkis, N.: Computing 2-isogenies between kummer lines. IACR Communications in Cryptology **1**(1) (2024). <https://doi.org/10.62056/abvua69p1>
45. Smith, B.A.: Explicit endomorphisms and correspondences. Ph.D. thesis (2005-12-23), <http://hdl.handle.net/2123/1066>
46. The Sage Developers: SageMath, the Sage Mathematics Software System (Version 10.0) (2023), <https://www.sagemath.org>
47. Van Wamelen, P.: Equations for the jacobian of a hyperelliptic curve. Transactions of the American Mathematical Society **350**(8), 3083–3106 (1998)



# SQIsign2D–West

## The Fast, the Small, and the Safer

Andrea Basso<sup>1,2</sup> , Pierrick Dartois<sup>3,4</sup> , Luca De Feo<sup>2</sup> ,  
Antonin Leroux<sup>5,6</sup> , Luciano Maino<sup>1</sup> , Giacomo Pope<sup>1,7</sup>,  
Damien Robert<sup>3,4</sup> , and Benjamin Wesolowski<sup>8</sup> 

<sup>1</sup> University of Bristol, Bristol, UK

`andrea.basso@bristol.ac.uk`

<sup>2</sup> IBM Research Europe, Zürich, Switzerland

<sup>3</sup> Univ. Bordeaux, CNRS, INRIA, IMB, UMR 5251, 33400 Talence, France

<sup>4</sup> INRIA, IMB, UMR 5251, 33400 Talence, France

<sup>5</sup> DGA-MI, Bruz, France

<sup>6</sup> IRMAR - UMR 6625, Université de Rennes, Rennes, France

<sup>7</sup> NCC Group, Cheltenham, UK

<sup>8</sup> ENS de Lyon, CNRS, UMPA, UMR 5669, Lyon, France

**Abstract.** We introduce SQIsign2D–West, a variant of SQIsign using two-dimensional isogeny representations.

SQIsignHD introduced four- and eight-dimensional isogeny representations to improve signing times and provable security of SQIsign, at the cost of slower verification. It left open the question of leveraging two-dimensional representations, which we solve here by introducing new algorithmic tools. These lead to a “best-of-both-worlds” scheme: our signing times are only  $2\times$  to  $3\times$  slower than SQIsignHD but  $10\times$  to  $15\times$  faster than SQIsign, our security proof rigorously reduces to an assumption similar to the one behind SQIsignHD, and our verification times are the fastest among all present variants of SQIsign. Additionally, like SQIsignHD, SQIsign2D–West favourably scales to high levels of security.

Concretely, for NIST level I we achieve signing times of 80 ms and verifying times of 4.5 ms, using optimised arithmetic for the `x86_64` architecture. For NIST level V, we achieve 470 ms for signing and 31 ms for verifying.

**Keywords:** Isogenies · Post-quantum · Signatures

## 1 Introduction

SQIsign [9, 14] is a signature scheme based on the conjectured hardness of computing endomorphism rings of supersingular curves. A candidate in the NIST post-quantum cryptography standardisation process, it features the smallest combined size of public key and signature, but it also exhibits one the slowest performances among all candidates.

The SIDH attacks [8, 30, 39] shook the foundations of isogeny-based cryptography by showing that any isogeny can be efficiently recovered from its evaluation on a sufficiently large torsion subgroup. Although they marked the end of

**Table 1.** Parameter sizes and performance of SQIsign2D–West. Average running times computed using an Intel Xeon Gold 6338 (Ice Lake, 2 GHz) using finite field arithmetic optimised for the x64 architecture, turbo boost disabled. See Sect. 6 for details.

	Sizes (bytes)		Timings (ms)		
	Public key	Signature	Keygen	Sign	Verify
NIST I	66	148	30	80	4.5
NIST III	98	222	85	230	14.5
NIST V	130	294	180	470	31.0

SIDH/SIKE [24, 25] and related schemes, it was not long before the same technique was put to constructive use, notably in the encryption schemes FESTA [4] and QFESTA [31], and in the SQIsignHD [11] variant of SQIsign. The key to all these applications is to *embed* an isogeny of elliptic curves into an isogeny between *higher-dimensional abelian varieties*. The number of dimensions used for the embedding is a key parameter for efficiency: Robert [38] shows that 8 dimensions are always enough, however the cost of representing the higher-dimensional objects grows *exponentially* with the dimension, thus all practical constructions strive to limit the embedding dimension. For example, FESTA and QFESTA manage to restrict themselves to two-dimensional isogenies.

In the same vein, SQIsignHD consists of two sub-variants. The first, Rigorous-SQIsignHD, uses eight-dimensional isogenies and strives for the best possible provable security but is deemed unpractical. The second, FastSQIsignHD, uses four-dimensional isogenies and compromises on the security proof to achieve the best possible efficiency: the result is a signature scheme with smaller signatures than SQIsign, similarly sized public keys, and significantly faster signing times, but, realistically, *slower verification* owing to the complexity of the four-dimensional representation.

**Our Contributions.** The question of whether it is possible to obtain an improvement over SQIsign by using only two-dimensional isogenies was left open in [11], where a short paragraph commented on the apparent difficulty of this task. We answer this question in the affirmative by introducing SQIsign2D–West.

To achieve this we introduce new tools for computing higher-dimensional isogeny representations in the context of supersingular elliptic curves:

- An algorithm, a simple extension of [31, Algorithm 2], to evaluate a random elliptic isogeny of given degree by embedding it in a two-dimensional isogeny;
- An algorithm, inspired by [34], to translate a quaternion ideal into a two-dimensional representation of the corresponding elliptic curve isogeny. Combined with an algorithm to sample uniformly random quaternion ideals of given norm, it lets the signer uniformly generate isogenies to be transmitted to the verifier.

We give concrete parametrisations of SQIsign2D–West for NIST security levels I, III and V, and implement them, using both generic and optimised modular

arithmetic. With key and signature sizes as reported in Table 1, it is comparable to SQIsignHD in terms of bandwidth. Our benchmarks highlight a consistent improvement over SQIsign across the whole spectrum, slightly slower signing performance than FastSQIsignHD but much faster than SQIsign, and *the fastest verification* among all variants of SQIsign. Because prime characteristics in the shape required by SQIsign2D–West are abundant, our variant, unlike SQIsign, does not need a costly search for a “SQIsign-friendly” prime and thus scales seamlessly to high security levels.

Our security proof shows that the security of SQIsign2D–West reduces to the problem of computing the endomorphism ring of a random supersingular curve, in a security model where the attacker is given (classical) access to an oracle computing (higher-dimensional representations of) uniformly random isogenies from a given curve. Hence, compared to SQIsignHD, SQIsign2D–West manages to blend the efficiency gains of FastSQIsignHD with security guarantees similar to RigorousSQIsignHD.

The algorithmic tools we introduce are very flexible, and we considered several variants with different trade offs between provable security and speed. In the main text, we focus on the most secure variant: our security proof follows the blueprint of RigorousSQIsignHD and achieves a reduction to the endomorphism ring problem, provided an isogeny-sampling oracle. By contrast the proof of unforgeability for SQIsign essentially assumes that the signing oracle does not leak information on the secrets. Nevertheless, if one is ready to accept heuristic security (roughly similar to the heuristics used in FastSQIsignHD, so still cleaner than the heuristics of SQIsign), it is possible to modify SQIsign2D–West to obtain even faster signing. We describe this variant in [3, Appendix B]

**Related Work.** Besides SQIsignHD, there is a growing interest in finding more efficient variants of SQIsign. The recent work *AprèsSQI* [41] achieves promising savings in verification, while keeping the general structure of SQIsign the same (in particular, *AprèsSQI* does not use higher dimensional isogenies). The key idea is to use larger extensions of the base field to access more small-order points of the curves, and thus more easy-to-compute isogenies. Nevertheless, because it does not change the overall structure, *AprèsSQI* suffers from the same problems as SQIsign when it comes to scaling: suitable primes are difficult to find and negatively impact the performance of high security levels.

While preparing this work, we were informed of three concurrent projects with similar objectives. What they have in common is the use of two-dimensional isogeny representations and prime characteristics of similar shape. In particular, they all scale to higher security levels more favourably than SQIsign. We briefly discuss the differences with our work below.

1. In [32], Nakagawa and Onuki first introduce an algorithm to translate ideals to isogenies relying on the computation of two-dimensional isogenies. This algo-

gorithm is reminiscent of the techniques used in [19]; in particular, it is significantly different from the one we introduce in Sect. 3.2. Then, they apply the algorithm to SQIsign. Their proof-of concept implementation in Julia suggests an improvement over SQIsign for key generation and signing, especially at higher security levels. The proof of security, however, remains heuristic.

2. In [33], Nakagawa and Onuki design SQIsign2D-East, a version of SQIsign where verification requires a computation of a two-dimensional isogeny. This idea shares many similarities with the heuristic version we describe in [3, Appendix B]. At the time of writing, we were not provided an implementation, but we expect SQIsign2D-East to have performance similar to our heuristic version. The main difference between this work and ours is the rigorous proof of security of SQIsign2D-West, which appears difficult to emulate with SQIsign2D-East.

Very recent work [7] shows that the version of SQIsign2D-East described in [33] did not reach the security levels claimed. The authors of [7] also present a new version of SQIsign2D-East that thwarts their attack. We highlight that this attack does not apply to SQIsign2D-West.

3. In [19], Duparc and Fouotsa introduce another version of SQIsign, called SQIPrime. SQIPrime is the closest to SQIsignHD of all the variants, the main difference being the type of challenge used in the identification protocol. The authors describe two versions, one using two-dimensional isogeny representations and another using four-dimensional ones. The security of either is close to FastSQIsignHD, and thus less rigorous than ours. No implementation of SQIPrime is available at the time, but we expect the four-dimensional variant to perform similarly to SQIsignHD, and the two-dimensional variant to perform similarly to SQIsign2D-East/West, albeit with larger keys and signatures.

For conciseness, from now on we will use SQIsign2D to refer to our protocol, only using SQIsign2D-West when it is needed to distinguish it from other variants.

**Plan.** We start by reviewing some mathematical background and the fundamentals of SQIsign and its variants in Sect. 2. In Sect. 3 we introduce our new algorithms to compute two-dimensional representations of isogenies. Building on these we give in Sect. 4 a detailed description of the SQIsign2D identification protocol, and provide a formal proof of its security in Sect. 5. Finally in Sect. 6 we describe our implementation of SQIsign2D-West and of its heuristic variant, and report on their performance. For space reasons, we describe the aforementioned heuristic variant in [3, Appendix A].

## 2 Preliminaries

In this section, we recall some background knowledge about the Deuring correspondence and isogenies between products of two elliptic curves. We assume some familiarity with elliptic curves and their isogenies and refer to [13, 42] for more information.

### 2.1 The Deuring Correspondence

We now give a brief summary of the theory of the Deuring correspondence, following the approach in [29, Chapter 2]. Let  $p > 3$  be a prime  $\equiv 3 \pmod{4}$  and let  $\mathcal{B}_{p,\infty}$  be the unique quaternion algebra ramified at  $p$  and  $\infty$ , i.e.  $\mathcal{B}_{p,\infty} = \mathbb{Q}\langle i, j \rangle$ , where  $i^2 = -1$  and  $j^2 = -p$ . Given a fractional ideal  $I$ , we define its left order as  $\mathcal{O}_L(I) = \{\alpha \in \mathcal{B}_{p,\infty} \mid \alpha I \subset I\}$ ; similarly, one can define its right order  $\mathcal{O}_R(I)$ .

In [17], Deuring showed an equivalence between maximal orders in  $\mathcal{B}_{p,\infty}$  and supersingular elliptic curves defined over  $\mathbb{F}_{p^2}$ . From now on, we will refer to this equivalence as the *Deuring correspondence*. Under this correspondence, an isogeny  $\varphi: E_1 \rightarrow E_2$  corresponds to a fractional ideal  $I_\varphi$ , where  $\mathcal{O}_L(I_\varphi) \cong \text{End}(E_1)$  and  $\mathcal{O}_R(I_\varphi) \cong \text{End}(E_2)$ . Moreover,  $\deg(\varphi) = \text{nr}(I_\varphi)$ .

Given two isogenies  $\varphi_1: E \rightarrow E_1$  and  $\varphi_2: E \rightarrow E_2$  of coprime degrees, we denote by  $[\varphi_1]_*\varphi_2: E_1 \rightarrow E'$  the pushforward isogeny of  $\varphi_2$  under  $\varphi_1$ , i.e.  $\ker([\varphi_1]_*\varphi_2) = \varphi_1(\ker(\varphi_2))$ . Equivalently, we define the pushforward of  $I_{\varphi_2}$  under  $I_{\varphi_1}$  as the ideal corresponding to the isogeny  $[\varphi_1]_*\varphi_2$ .

A problem we will face in the following sections is to compute the ideal associated to a given kernel generator. To be more precise, we are given an isogeny  $\varphi: E_0 \rightarrow E$ , where we know  $\mathcal{O}_0 \cong \text{End}(E_0)$  and its associated ideal  $I_\varphi$ . We also have a point  $K \in E$  of smooth order  $D$  coprime to  $\deg(\varphi)$ , which describes the kernel of an isogeny  $\psi: E \rightarrow E'$ . Our goal is to compute  $I_\psi$ , the ideal corresponding to  $\psi$ .

We can accomplish this goal using the algorithm [11, Algorithm 9]. In particular, we first push  $\mathcal{O}_0$  under  $\varphi$  via [11, Algorithm 8] and then use [11, Algorithm 9] to retrieve  $I_\psi$ . In our use case, we want to avoid running [11, Algorithm 8] and [11, Algorithm 9] on the fly but rather allow some precomputations. Let  $(P, Q)$  be a basis  $E[D]$  and write  $K$  as  $[a]P + [b]Q$ . In [11, Algorithm 9, Line 1], we have to evaluate a basis  $(\beta_1, \beta_2, \beta_3, \beta_4)$  of the right order of  $I_\varphi$  at  $K$ . This is equivalent to evaluating  $(\beta_1, \beta_2, \beta_3, \beta_4)$  at the basis  $(P, Q)$  and then retrieving  $\beta_i(K)$  as  $[a]\beta_i(P) + [b]\beta_i(Q)$ .

In what follows, we use the notation  $\{\beta_i(P), \beta_i(Q)\}_{i=1,\dots,4}$  to mean that we have evaluated a basis  $(\beta_1, \beta_2, \beta_3, \beta_4)$  of the right order of  $I_\varphi$  at  $(P, Q)$  via [11, Algorithm 9]. Additionally, we say that we use the datum  $\{\beta_i(P), \beta_i(Q)\}_{i=1,\dots,4}$  to compute  $I_\psi$  when we evaluate  $(\beta_1, \beta_2, \beta_3, \beta_4)$  at  $K$  as  $([a]\beta_i(P) + [b]\beta_i(Q))_{i=1,\dots,4}$  and then run the rest of [11, Algorithm 9] to obtain  $I_\psi$ .

### 2.2 Kani’s Lemma

Throughout this document we will encounter several different ways to represent isogenies of elliptic curves. We abstract the details into the concept of *isogeny representation*, which essentially says that representing an isogeny is having an efficient algorithm to evaluate it.

**Definition 1.** Let  $\mathbb{F}_q$  be a finite field. An isogeny evaluator  $\mathcal{E}$  is a pair of polynomial-time algorithms:

- $\mathcal{E}.\text{valid}(D)$  taking as input a string  $D \in \{0, 1\}^*$  and outputting either a symbol  $\perp$  or a triple  $(E, E', d)$ ; in the latter case,  $E$  and  $E'$  are elliptic curves defined over  $\mathbb{F}_q$  and there exists an isogeny  $\varphi : E \rightarrow E'$  of degree  $d$ .
- $\mathcal{E}.\text{eval}(D, P)$  taking as input a string  $D \in \{0, 1\}^*$  and a point  $P \in E(\mathbb{F}_{q^k})$ ; if  $\mathcal{E}.\text{valid}(D) = (E, E', d)$  it outputs the image point  $\varphi(P) \in E'(\mathbb{F}_{q^k})$ , otherwise the output is undefined.

In the case that  $D$  is of size polynomial in  $\log(q)$  and  $\log(d)$  and that  $\mathcal{E}.\text{valid}(D)$  does not output  $\perp$ , the string  $D$  is called an efficient representation of  $\varphi$  (for the evaluator  $\mathcal{E}$ ).

The article [38] shows that any isogeny can be efficiently represented as the datum of its evaluation on a suitably chosen set of points, then gives an efficient algorithm, akin to an *interpolation-evaluation* algorithm, which, on input an arbitrary point  $x$  and the evaluation datum, outputs the value of the isogeny at  $x$ . We will only need a special case of this construction, embedding an arbitrary dimension-one  $n$ -isogeny into a two-dimensional  $2^e$ -isogeny where  $2^e > n$ . Let us first recall the notion of  $(d_1, d_2)$ -isogeny diamond.

**Definition 2.** A  $(d_1, d_2)$ -isogeny diamond is a commutative diagram of isogenies:

$$\begin{array}{ccc}
 E_0 & \xrightarrow{\varphi_1} & E_1 \\
 \varphi_2 \downarrow & \circlearrowleft & \downarrow \varphi'_2 \\
 E_2 & \xrightarrow{\varphi'_1} & E_{12}
 \end{array}$$

where  $\varphi_1 : E_0 \rightarrow E_1$  and  $\varphi'_1 : E_2 \rightarrow E_{12}$  are  $d_1$ -isogenies,  $\varphi_2 : E_0 \rightarrow E_2$  and  $\varphi'_2 : E_1 \rightarrow E_{12}$  are  $d_2$ -isogenies.

We can now state Kani’s Lemma, which is contained in [26, Section 2, Proof of Theorem 2.3]. We refer to [30, Theorem 1] for a proof of this result.

**Theorem 4 (Kani’s Lemma).** Let  $d_1$  and  $d_2$  be two coprime positive integers. Given a  $(d_1, d_2)$ -isogeny diamond, the isogeny  $\Phi : E_0 \times E_{12} \rightarrow E_1 \times E_2$  given matrixially by

$$\Phi = \begin{pmatrix} \varphi_1 & \hat{\varphi}'_2 \\ -\varphi_2 & \hat{\varphi}'_1 \end{pmatrix}$$

is a  $(d_1 + d_2)$ -isogeny between these products of elliptic curves with their principal product polarisation. The kernel of  $\Phi$  is given by

$$\text{Ker } \Phi = \{(\hat{\varphi}_1(P), \varphi'_2(P)) \mid P \in E_1[d_1 + d_2]\}.$$

### 2.3 The SQIsign family

**SQIsign and SQIsignHD.** SQIsign is a digital signature scheme obtained via the Fiat-Shamir transform [21] of an identification protocol. This protocol is built on the Deuring correspondence between quaternion ideals and isogenies. SQIsign and SQIsignHD mainly differ in the way of making the Deuring correspondence effective. While SQIsign only works with smooth degree isogenies between supersingular elliptic curves, SQIsignHD uses four-dimensional isogenies in the verification process. In the following, we present the main building blocks of SQIsign (and SQIsignHD) identification protocol which will be used in SQIsign2D.

*Public Set-Up.* We choose a prime  $p$  and a supersingular elliptic curve  $E_0/\mathbb{F}_{p^2}$  of known endomorphism ring  $\mathcal{O}_0 \cong \text{End}(E_0)$  such that  $E_0$  has smooth torsion defined over a small extension of  $\mathbb{F}_{p^2}$  (of degree 1 or 2). In practice, one may use the curve  $E_0 : y^2 = x^3 + x$  (and  $p \equiv 3 \pmod{4}$ ).

*Key Generation.* The prover generates a random secret isogeny  $\varphi_{\text{sk}} : E_0 \rightarrow E_{\text{pk}}$  and publishes  $E_{\text{pk}}$  as its public key.

*Commitment.* The prover generates a random secret isogeny  $\varphi_{\text{com}} : E_0 \rightarrow E_{\text{com}}$  and sends  $E_{\text{com}}$  to the verifier as its commitment. For the identification protocol to be zero-knowledge (and the derived signature scheme to be secure),  $E_{\text{com}}$  has to be computationally indistinguishable from a uniformly random elliptic curve in the supersingular isogeny graph.

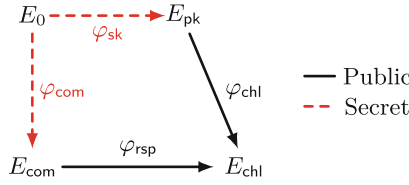
*Challenge.* The verifier generates and sends to the prover a random isogeny  $\varphi_{\text{chl}} : E_{\text{pk}} \rightarrow E_{\text{chl}}$  of smooth degree sufficiently large for  $\varphi$  to have high entropy. The challenge space should have size  $\Omega(2^\lambda)$  to ensure  $\lambda$  bits of (soundness) security.

*Response.* The prover generates and transmits to the verifier an *efficient representation* (as defined in Definition 1) of an isogeny  $\varphi_{\text{rsp}} : E_{\text{com}} \rightarrow E_{\text{chl}}$  which does not backtrack through  $\varphi_{\text{chl}}$  (i.e.  $\hat{\varphi}_{\text{rsp}} \circ \varphi_{\text{chl}}$  is cyclic).

*Verification.* The verifier checks that the response returned by the prover correctly represents an isogeny  $\varphi_{\text{rsp}} : E_{\text{com}} \rightarrow E_{\text{chl}}$  and checks that this isogeny does not backtrack through  $\varphi_{\text{chl}}$ . The diagram in Fig. 1 illustrates the relationship between the various isogenies computed by the protocol.

To compute such an efficient representation of  $\varphi_{\text{rsp}}$  (that will be called  $\varphi_{\text{rsp}}$  by abuse of notations), the prover uses the Deuring correspondence. Returning  $\varphi_{\text{rsp}} = \varphi_{\text{chl}} \circ \varphi_{\text{sk}} \circ \hat{\varphi}_{\text{com}} : E_{\text{com}} \rightarrow E_{\text{chl}}$  would make the scheme insecure. However, the prover can translate  $\varphi_{\text{chl}} \circ \varphi_{\text{sk}} \circ \hat{\varphi}_{\text{com}}$  into an ideal  $I$  connecting  $\text{End}(E_{\text{com}})$  and  $\text{End}(E_{\text{chl}})$ , find a random equivalent ideal  $I_{\text{rsp}} \sim I$  and translate  $I_{\text{rsp}}$  into  $\varphi_{\text{rsp}}$ .





**Fig. 1.** The SQIsign/SQIsignHD identification protocol. Dashed red lines represent secrets. (Color figure online)

The ideal  $I_{rsp} \sim I$  is sampled to be relatively easy to translate into an isogeny and with a distribution that ensures one can simulate the response without secret knowledge (zero knowledge property). Those two objectives are in tension and lead to a trade-off between efficiency and rigorous security proofs. In SQIsign,  $\text{nrd}(I_{rsp})$  had to be smooth to make the ideal to isogeny translation possible. The KLPT algorithm [28] was used to find  $I_{rsp}$ , resulting in big norms  $\text{nrd}(I_{rsp}) \approx p^{15/4}$ , slow ideal to isogeny translation and a very heuristic security proof.

In SQIsignHD [11], the smoothness condition on  $I_{rsp}$  is relaxed, allowing for smaller norms, a stronger security proof and a faster response at the expense of the verification time. The idea is to use the higher-dimensional SIDH attack techniques [8, 30, 39] to represent  $\varphi_{rsp}$ . The prover uses the secret knowledge of  $\varphi_{chl} \circ \varphi_{sk} \circ \widehat{\varphi}_{com}$  to evaluate  $\varphi_{rsp}$  on some torsion points. This torsion evaluation (along with  $\text{deg}(\varphi_{rsp})$ ) is an efficient representation of  $\varphi_{rsp}$  that can be sent to the verifier. To verify the validity of this representation, the verifier computes a four-dimensional isogeny that “embeds”  $\varphi_{rsp}$  by Kani’s Lemma. The efficiency of four-dimensional isogeny computation is still an open research question. However, SQIsignHD verification is expected to be slower than SQIsign verification, especially after the latest improvements of AprèsSQI [41]. This was the main motivation for our contribution: accelerate the verification while maintaining a fast signing procedure and strong security proofs (with two-dimensional isogeny computations).

**What Makes SQIsign2D Different from SQIsign and SQIsignHD.** As a derivative of SQIsign, SQIsign2D follows the same construction presented above but uses different techniques involving two-dimensional isogeny computations. To perform the verification, we “embed” the response  $\varphi_{rsp} : E_{com} \rightarrow E_{chl}$  into a two-dimensional  $2^r$ -isogeny. The bottleneck is to find an auxiliary isogeny  $\varphi_{aux} : E_{chl} \rightarrow E_{aux}$  of degree  $2^r - \text{deg}(\varphi_{rsp})$  to complete the isogeny diamond and apply Kani’s Lemma. Additionally, the distribution of  $\varphi_{aux}$  needs to be uniform in order to simplify the proof of the zero knowledge property.

We overcome these issues with an algorithm to sample quaternion ideals of fixed norm with a uniform distribution (called `RandomFixedNormIdeal`) and another algorithm (called `IdealToIsogeny`) to translate any left ideal of the order  $\mathcal{O}_0 \cong \text{End}(E_0)$  (with  $j(E_0) = 1728$ ) into an isogeny. `IdealToIsogeny` uses two-dimensional isogenies and is inspired from the Clapoti algorithm introduced in [34] and `RandIsoglImages` introduced in QFESTA [31, Algorithm 2]. Both `RandomFixedNormIdeal` and `IdealToIsogeny` are also used in the key generation and

commitment steps to obtain statistically uniform distributions of  $E_{pk}$  and  $E_{com}$ , with a clear security benefit.

### 3 Algorithmic Building Blocks

In this section, we present the main algorithmic building blocks of SQIsign-2D to make the Deuring correspondence effective. We assume we are given a cryptographic size prime  $p = c \cdot 2^e - 1$  with  $e \in \mathbb{N}$  and  $c \in \mathbb{N}$  as small as possible. We can find such  $p$  with  $c = O(\log(p))$  by Dirichlet’s arithmetic progression theorem [18]. We denote by  $E_0$  the supersingular elliptic curve given by  $y^2 = x^3 + x$  over  $\mathbb{F}_p$  and by  $\mathcal{O}_0$  a maximal quaternion order isomorphic to  $\text{End}(E_0)$ .

First, we briefly introduce `FixedDegreelsogeny`, an algorithm to compute the kernel ideal and to evaluate an isogeny of fixed odd degree defined over  $E_0$ , which is almost identical to `RandsogImages` introduced in QFESTA [31, Algorithm 2]. Then, we present an algorithm `IdealTolsogeny` to translate any left ideal of  $\mathcal{O}_0$  into an efficient representation of isogeny defined over  $E_0$ . We finally present an algorithm `RandomFixedNormIdeal` to sample left ideals of a given maximal order  $\mathcal{O} \subseteq \mathcal{B}_{p,\infty}$  of fixed norm with a uniform distribution.

#### 3.1 Generating an Arbitrary Odd-Degree Isogeny from $E_0$

In QFESTA [31], Nakagawa and Onuki introduce an algorithm `RandsogImages` to compute non-smooth isogenies originating from  $E_0$ . For SQIsign2D, we use their idea and tweak it to construct the `FixedDegreelsogeny` algorithm which:

- Takes as input an odd positive integer  $u < 2^e$  and a basis  $(P_0, Q_0)$  of  $E_0[2^e]$ .
- Returns the torsion image points  $\varphi|_{2^e} = (\varphi(P_0), \varphi(Q_0))$  and the codomain  $E$ , where  $\varphi: E_0 \rightarrow E$  is a  $u$ -isogeny (as in `RandsogImages`), along with its corresponding ideal  $I$  (not returned by `RandsogImages`).

In the rest of the paper, we will use the notation  $\varphi|_N$  to refer to the action of  $\varphi$  on  $E_0[N]$ . In practice, when we write  $\varphi|_N$ , we mean  $\varphi(P)$  and  $\varphi(Q)$ , for some basis  $\langle P, Q \rangle = E_0[N]$  (as above). A detailed description of `FixedDegreelsogeny` is provided in [3, Appendix A.1]. It involves Kani’s Lemma and the computation of a  $2^e$ -isogeny.

#### 3.2 Translating a Left Ideal Into an Efficient Isogeny Representation

The state of the art techniques to translate ideals into isogenies impose conditions on the input norm. In SQIsign, the norm had to be smooth and in SQIsignHD, the norm  $\text{nrd}(I)$  had to be such that  $2^e - \text{nrd}(I)$  can be easily decomposed into a sum of two squares. We now propose an algorithm `IdealTolsogeny` to translate a left  $\mathcal{O}_0$ -ideal  $I$  of any norm into an isogeny starting from  $E_0$ . It is inspired by Page and Robert’s work in the context of the Clapoti group action [34]. In Clapoti, the ideal considered is an ideal of a quadratic imaginary order but we can adapt their ideas to quaternion orders.

Let  $I$  be a left  $\mathcal{O}_0$ -ideal. We want to compute the torsion image  $\varphi_I|_{2^e}$ . The general outline is as follows:

1. Find  $I_1, I_2 \sim I$  of coprime norms  $d_1, d_2 \approx \sqrt{p}$ , and  $u, v \in \mathbb{N}^*$  such that  $d_1u + d_2v = 2^f$  with  $f \leq e$  and  $d_1u$  is prime to  $d_2v$ .
2. Evaluate isogenies  $\varphi_u, \varphi_v : E_0 \rightarrow E_u, E_v$  of degrees  $u$  and  $v$  on  $E_0[2^e]$ .
3. Use Kani's Lemma on  $\varphi_u \circ \widehat{\varphi}_1 : E_I \rightarrow E_u$  and  $\varphi_v \circ \widehat{\varphi}_2 : E_I \rightarrow E_v$ , where  $\varphi_1, \varphi_2 : E_0 \rightarrow E_I$  are the isogenies corresponding to  $I_1$  and  $I_2$  respectively, to compute  $\Phi : E_u \times E_v \rightarrow E_I \times E'$  that embeds the isogenies  $\varphi_1 \circ \widehat{\varphi}_u$  and  $\varphi_2 \circ \widehat{\varphi}_v$ .
4. Use  $\Phi$  to compute  $\varphi_1 \circ \widehat{\varphi}_u|_{2^e}$  and then  $\varphi_u|_{2^e}$  to obtain  $\varphi_1|_{2^e}$  and finally obtain  $\varphi_I|_{2^e}$ .

**Step 1.** We sample ideals  $I_1, I_2 \sim I$  of odd coprime norms  $d_1$  and  $d_2$  until we find positive integers  $u, v$  such that  $d_1u + d_2v = 2^e$ . A sufficient (but not necessary) condition for a solution  $(u, v)$  to exist is  $d_1d_2 < 2^e$ . Hence, the norms  $d_1$  and  $d_2$  should be as small as possible. To find equivalent ideals of such norms, we sample  $\beta_i \in I$  with sufficiently small reduced norm and choose  $I_i := I\beta_i/\text{nrd}(I)$ , so that  $\text{nrd}(I_i) = q_I(\beta_i) := \text{nrd}(\beta_i)/\text{nrd}(I)$ . Minkowski's theorem and [28, Section 3.1] (see also [11, Lemma 12]) ensure that the shortest vector in  $I$  has norm  $O(\text{nrd}(I)\sqrt{p})$  so we should expect to find  $d_1, d_2 \approx \sqrt{p}$  so that  $d_1d_2 \approx p \approx 2^e$  in general. This is not enough to rigorously ensure the existence of  $u$  and  $v$ .

In [3, Section 3.1], we provide an algorithm ([3, Algorithm 1]) which samples  $\beta_1, \beta_2 \in I$  and finds  $u, v \in \mathbb{N}^*$  such that  $\text{gcd}(uq_I(\beta_1), vq_I(\beta_2)) = 1$  and  $uq_I(\beta_1) + vq_I(\beta_2) = 2^e$ , where  $q_I(\beta) := \text{nrd}(\beta)/\text{nrd}(I)$ . This algorithm terminates after  $O(\log(p)^2)$  attempts (to sample  $\beta_1, \beta_2$ ) under reasonable heuristics that we motivate therein.

**Step 2.** We can use FixedDegreeIsogeny [3, Algorithm 7] to evaluate isogenies  $\varphi_u, \varphi_v : E_0 \rightarrow E_u, E_v$  of degrees  $u$  and  $v$  on  $E_0[2^e]$ . Since  $u, v \approx \sqrt{p}$ , we do not need to compute two-dimensional 2-isogeny chains of full length  $e$  in this step, but of half length  $e/2$  instead (see [3, Remark 26]).

*Remark 8.* Alternatively, we may save some time on step 2 at the expense of step 1. Assuming  $u = a^2 + b^2$ , with  $a, b \in \mathbb{Z}$ , then we can choose  $\varphi_u := [a] + [b]\iota \in \text{End}(E_0)$ , with  $\iota : (x, y) \in E_0 \mapsto (-x, \sqrt{-1}y) \in E_0$  and similarly for  $v$ . Finding  $u, v$  in step 1 that can be written easily as a sum of squares is more costly. There is also a hybrid approach where we only require  $u$  (or  $v$ ) to be a sum of two squares. Experimentally, both of these approaches were on the whole more costly than the proposed method as soon as the ideal given in input is a bit unbalanced (and the smallest possible  $d_2$  is a bit bigger than the expected  $\approx \sqrt{p}$ ). However, we believe that there is room for improvement in our implementation of this search for  $d_1, d_2, u$  and  $v$ , and this could lead to a different conclusion regarding which variant is the most efficient. Answering this interrogation is left as an interesting open question for future work.

**Step 3.** We now give more details on steps 3 and 4 inspired by [34]. Consider the following  $(d_1u, d_2v)$ -isogeny diamond:

$$\begin{array}{ccc}
 E' & \xrightarrow{\widehat{\varphi}'_v} & E_v \\
 \varphi'_u \uparrow & \circlearrowleft & \uparrow \varphi_v \circ \widehat{\varphi}_2 \\
 E_u & \xrightarrow{\varphi_1 \circ \widehat{\varphi}_u} & E_I
 \end{array}$$

where  $\varphi'_u := [\varphi_u \circ \widehat{\varphi}_1]_*(\varphi_v \circ \widehat{\varphi}_2)$  and  $\varphi'_v := [\varphi_v \circ \widehat{\varphi}_2]_*(\varphi_u \circ \widehat{\varphi}_1)$  (pushforward isogenies). By Kani’s Lemma, we have a  $2^f$ -isogeny:

$$\Phi := \begin{pmatrix} \varphi_1 \circ \widehat{\varphi}_u & \varphi_2 \circ \widehat{\varphi}_v \\ -\varphi'_u & \varphi'_v \end{pmatrix} : E_u \times E_v \rightarrow E_I \times E',$$

with kernel:

$$\ker(\Phi) = \{([d_1]\varphi_u(P), \varphi_v \circ \widehat{\varphi}_2 \circ \varphi_1(P)) \mid P \in E_0[2^f]\}.$$

Let  $\theta := \widehat{\varphi}_2 \circ \varphi_1 \in \text{End}(E_0)$ . By Lemma 9, given  $I_1$  and  $I_2$ , if we write  $I_1 := I\overline{\beta}_1/\text{nrd}(I)$  and  $I_2 := I\overline{\beta}_2/\text{nrd}(I)$  with  $\beta_1, \beta_2 \in I$ , then we can compute  $\theta = \beta_2\overline{\beta}_1/\text{nrd}(I)$  so we can evaluate it easily. By step 2, we also know  $\varphi_v|_{2^e}$  and  $\varphi_u|_{2^e}$ . Hence, we can compute  $\ker(\Phi)$  (and evaluate  $\Phi$ ) efficiently. This completes step 3.

**Step 4.** We first notice that we can evaluate  $\varphi_1 \circ \widehat{\varphi}_u$  from the two-dimensional isogeny  $\Phi$ . This implies we can evaluate  $\varphi_1$  on  $E_0[2^e]$  as follows:  $\Phi(\varphi_u(P_0), 0) = ([u]\varphi_1(P_0), *)$  and  $\Phi(\varphi_u(Q_0), 0) = ([u]\varphi_1(Q_0), *)$  and we can invert  $u$  modulo  $2^e$  since  $u$  is odd to get  $\varphi_1|_{2^e} = (\varphi_1(P_0), \varphi_1(Q_0))$ . To obtain  $\varphi_I|_{2^e}$ , we rely on the following lemma.

**Lemma 9.** For  $i \in \{1, 2\}$ , if we write  $I_i := I\overline{\beta}_i/\text{nrd}(I)$  with  $\beta_i \in I$ , then  $\widehat{\varphi}_i \circ \varphi_I = \beta_i$ .

*Proof.* Let  $i \in \{1, 2\}$ . We should expect that  $\widehat{\varphi}_i \circ \varphi_I$  corresponds to the ideal  $I \cdot \overline{I}_i$ , however  $\mathcal{O}_R(I) \neq \mathcal{O}_L(\overline{I}_i)$  so the product  $I \cdot \overline{I}_i$  is not well defined. It is defined up to conjugation of  $\overline{I}_i$ . We have  $\mathcal{O}_L(\overline{I}_i) = \mathcal{O}_R(I_i) = \overline{\beta}_i^{-1}\mathcal{O}_R(I)\overline{\beta}_i$ . It follows that  $\mathcal{O}_L(\overline{\beta}_i \cdot \overline{I}_i \cdot \overline{\beta}_i^{-1}) = \mathcal{O}_R(I)$  and the ideal corresponding to the isogeny  $\widehat{\varphi}_i \circ \varphi_I$  via the Deuring correspondence is:

$$I\overline{\beta}_i \cdot \overline{I}_i \cdot \overline{\beta}_i^{-1} = I \frac{\overline{\beta}_i\beta_i}{\text{nrd}(I)} \overline{I} \cdot \overline{\beta}_i^{-1} = I\overline{I} \frac{\text{nrd}(\beta_i)}{\text{nrd}(I)} \frac{\beta_i}{\text{nrd}(\beta_i)} = \mathcal{O}_0\beta_i.$$

The result follows. □

Following Lemma 9, we have that  $[d_1]\varphi_I = \varphi_1 \circ \beta_1$ . Since we can evaluate  $\beta_1$  and  $\varphi_1$  on  $E_0[2^e]$  and  $d_1$  can be inverted modulo  $2^e$ , we can evaluate  $\varphi_I$  on  $E_0[2^e]$ , completing step 4. Algorithm 2 summarises all these steps.

---

**Algorithm 2.** IdealTolsogeny

---

**Input:** An ideal  $I \subseteq \mathcal{O}_0 \cong \text{End}(E_0)$  and a basis  $(P_0, Q_0)$  of  $E_0[2^e]$ .

**Output:** The image  $\varphi_I|_{2^e} = (\varphi_I(P_0), \varphi_I(Q_0))$  of the isogeny  $\varphi_I : E_0 \rightarrow E_I$  associated to  $I$ .

- 1: Use [3, Algorithm 1] to obtain  $\beta_1, \beta_2 \in I$  and  $u, v \in \mathbb{N}^*$  and  $f \leq e$  such that  $\gcd(uq_I(\beta_1), vq_I(\beta_2)) = 1$  and  $uq_I(\beta_1) + vq_I(\beta_2) = 2^f$
  - 2:  $I_i \leftarrow I\beta_i / \text{nrd}(I)$  for  $i \in \{1, 2\}$
  - 3:  $\theta \leftarrow \beta_2\bar{\beta}_1 / \text{nrd}(I) \in \text{End}(E_0)$  ( $\triangleright$ )  $\theta := \widehat{\varphi}_2 \circ \varphi_1$
  - 4: Compute  $\varphi_u|_{2^e}$  for a  $u$ -isogeny  $\varphi_u : E_0 \rightarrow E_u$  ( $\triangleright$ ) FixedDegreelsogeny( $u, P_0, Q_0$ )
  - 5: Compute  $\varphi_v|_{2^e}$  for a  $v$ -isogeny  $\varphi_v : E_0 \rightarrow E_v$  ( $\triangleright$ ) FixedDegreelsogeny( $v, P_0, Q_0$ )
  - 6: Set  $K_P \leftarrow [2^{e-f}]([d_1]\varphi_u(P_0), \varphi_v \circ \theta(P_0))$
  - 7: Set  $K_Q \leftarrow [2^{e-f}]([d_1]\varphi_u(Q_0), \varphi_v \circ \theta(Q_0))$
  - 8: Compute  $\Phi : E_u \times E_v \rightarrow E_I \times E'$  of kernel  $\langle K_P, K_Q \rangle$
  - 9: Evaluate  $\Phi(\varphi_u(P_0), 0) = ([u]\varphi_1(P_0), *)$  and  $\Phi(\varphi_u(Q_0), 0) = ([u]\varphi_1(Q_0), *)$  to obtain  $\varphi_1|_{2^e}$
  - 10: Use  $\varphi_1|_{2^e}$  to evaluate  $\varphi_I = [1/d_1]\varphi_1 \circ \beta_1$  on  $(P_0, Q_0)$  and obtain  $\varphi_I|_{2^e}$
  - 11: **return**  $\varphi_I|_{2^e}$
- 

**3.3 Sampling Uniformly at Random an Ideal of Fixed Norm**

In the protocol, we shall need to uniformly sample at random cyclic isogenies  $\varphi : E \rightarrow E'$  of fixed degree  $N$  several times. When  $\mathcal{O} \cong \text{End}(E)$  is known, by the Deuring correspondence this reduces to sampling a left ideal  $I \subseteq \mathcal{O}$  of norm  $N$  uniformly at random.  $I$  is then translated into an isogeny  $\varphi$  (e.g. using Algorithm 2 if  $\mathcal{O} = \mathcal{O}_0$ ). For  $\varphi$  to be cyclic,  $I$  has to be *primitive*, that is to say that  $I \not\subseteq n\mathcal{O}$  for any integer  $n > 1$ .

Given a maximal quaternion order  $\mathcal{O} \subseteq \mathcal{B}_{p,\infty}$  and an integer  $N$  coprime with  $p$ , we explain how to sample primitive left ideals  $I \subseteq \mathcal{O}$  of norm  $N$ . It has been proved that such ideals are in bijection with primitive left ideals of  $\mathcal{O}/N\mathcal{O}$  via the reduction modulo  $N$  which are themselves in bijection with:

$$\mathbb{P}^1(\mathbb{Z}/N\mathbb{Z}) = \{(x, y) \in (\mathbb{Z}/N\mathbb{Z})^2 \mid \gcd(x, y, N) = 1\} / (\mathbb{Z}/N\mathbb{Z})^*$$

$N$  being coprime with  $p$ ,  $\mathcal{B}_{p,\infty}$  splits at  $N$  and we have an isomorphism  $\mathcal{O} \otimes \mathbb{Z}_N \cong M_2(\mathbb{Z}_N)$ , where  $\mathbb{Z}_N$  is the completion of the localisation of  $\mathbb{Z}$  at  $N$ . Via the reduction modulo  $N$ , we obtain an isomorphism  $\varphi_N : \mathcal{O}/N\mathcal{O} \xrightarrow{\sim} M_2(\mathbb{Z}/N\mathbb{Z})$ .

**Lemma 10** ([27, Lemma 7.2]). *All primitive left ideals of  $M_2(\mathbb{Z}/N\mathbb{Z})$  are principal and generated by a matrix*

$$M_{x,y} = \begin{pmatrix} x & y \\ 0 & 0 \end{pmatrix}$$

with  $(x : y) \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$ . Hence, we have the following bijection:

$$\begin{aligned} \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z}) &\longrightarrow \{\text{primitive left ideals } I \subseteq \mathcal{O} \text{ of norm } N\} \\ (x : y) &\longmapsto \mathcal{O}\varphi_N^{-1}(M_{x,y}) + \mathcal{O}N \end{aligned}$$

As a direct consequence of the above lemma, we obtain:

**Lemma 11.** *The set of elements  $\alpha \in \mathcal{O}$  invertible modulo  $N$  acts transitively (by multiplication on the right) on the set of primitive left  $\mathcal{O}$ -ideals of norm  $N$ . Those elements  $\alpha \in \mathcal{O}$  invertible modulo  $N$  are those of norm coprime with  $N$ .*

*Proof.* Let  $I$  be a primitive left  $\mathcal{O}$ -ideal of norm  $N$ . Then, the ideal  $I$  corresponds to  $(x : y) \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$  via the bijection of Lemma 10 and is isomorphic to  $M_2(\mathbb{Z}/N\mathbb{Z}) \cdot M_{x,y}$  via the composition of the reduction modulo  $N$  and  $\varphi_N$ . For any representative  $(x, y) \in \mathbb{Z}^2$  of  $(x : y) \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$ , we have  $\gcd(x, y, N) = 1$  so we may find  $u, v \in \mathbb{Z}$  such that  $xu + yv \equiv 1 \pmod N$ , so that:

$$M_{x,y} \begin{pmatrix} u & -y \\ v & x \end{pmatrix} \equiv M_{1,0} \pmod N \quad \text{and} \quad \det \begin{pmatrix} u & -y \\ v & x \end{pmatrix} \equiv 1 \pmod N$$

Hence, the ideal  $M_2(\mathbb{Z}/N\mathbb{Z}) \cdot M_{x,y}$  is in the orbit of  $M_2(\mathbb{Z}/N\mathbb{Z}) \cdot M_{1,0}$  under the right action of  $GL_2(\mathbb{Z}/N\mathbb{Z})$ , and as a consequence,  $I/N\mathcal{O}$  is in the orbit of the ideal  $I_0/N\mathcal{O} := \mathcal{O}\varphi_N^{-1}(M_{1,0})/N\mathcal{O}$  under the right action of  $(\mathcal{O}/N\mathcal{O})^*$ .

To conclude, it suffices to prove that the invertible elements of  $\mathcal{O}$  modulo  $N$  are those of norm coprime with  $N$ . If  $\alpha \in \mathcal{O}$  is invertible modulo  $N$ , there exists  $\beta, \gamma \in \mathcal{O}$  such that  $\alpha\beta = 1 + N\gamma$ , so that

$$\text{nrd}(\alpha) \text{nrd}(\beta) = \text{nrd}(1 + N\gamma) = 1 + N \text{Tr}(\gamma) + N^2 \text{nrd}(\gamma) \equiv 1 \pmod N,$$

so  $\text{nrd}(\alpha)$  is invertible modulo  $N$ . Conversely, if  $\text{nrd}(\alpha)$  is prime to  $N$ , there exists  $\lambda \in \mathbb{Z}$  such that  $\text{nrd}(\alpha)\lambda \equiv 1 \pmod N$ . Then, it follows that  $\alpha\bar{\alpha}\lambda \equiv 1 \pmod N$ , so  $\alpha$  is invertible modulo  $N$ . This completes the proof.  $\square$

Lemma 11 ensures that  $(\mathcal{O}/N\mathcal{O})^*$  acts transitively on primitive left ideals of norm  $N$  by multiplication on the right. Hence, given a primitive left  $\mathcal{O}$ -ideal  $I_0$  of norm  $N$ , if we sample  $[\alpha] \in (\mathcal{O}/N\mathcal{O})^*$  uniformly at random, then  $I_0\alpha + N\mathcal{O}$  is uniformly random among primitive left  $\mathcal{O}$ -ideals of norm  $N$ .

To obtain such an ideal  $I_0$ , we compute  $\gamma \in \mathcal{O}$  of norm  $NM$  with  $\gcd(N, M) = 1$  and without integral factor. This can be done with the algorithms of [29, Section 3.3]. We then consider  $I_0 := \mathcal{O}\gamma + \mathcal{O}N$  and sample  $[\alpha] \in \mathcal{O}/N\mathcal{O}$  uniformly at random until it is invertible modulo  $N$  (which can be checked by computing  $\text{nrd}(\alpha)$ ). The probability of finding such an  $\alpha$  is (by the Chinese remainder theorem):

$$\frac{|GL_2(\mathbb{Z}/N\mathbb{Z})|}{|M_2(\mathbb{Z}/N\mathbb{Z})|} = \prod_{\ell^e \parallel N} \frac{|GL_2(\mathbb{Z}/\ell^e\mathbb{Z})|}{|M_2(\mathbb{Z}/\ell^e\mathbb{Z})|} = \prod_{\ell \mid N} \left(1 - \frac{1}{\ell}\right) \left(1 - \frac{1}{\ell^2}\right).$$

This quantity is an  $\Omega(1/\log \log(N))$  by [23, Theorem 328] so we can find  $\alpha$  after  $O(\log \log(N))$  tries. These operations are summarised in Algorithm 3.

---

**Algorithm 3.** RandomFixedNormIdeal

---

**Input:** A maximal order  $\mathcal{O} \subseteq \mathcal{B}_{p,\infty}$  and an integer  $N$  such that  $p \nmid N$ .

**Output:** A primitive left  $\mathcal{O}$ -ideal  $I$  of norm  $N$  sampled uniformly at random.

- 1: Find  $\gamma \in \mathcal{O}$  primitive of norm  $NM$  with  $\gcd(N, M) = 1$  ( $\triangleright$ ) Using [29, Section 3.3]
  - 2: **repeat**
  - 3:   Sample  $u_1, \dots, u_4 \in \llbracket 0; N - 1 \rrbracket$  uniformly at random
  - 4:    $\alpha \leftarrow \sum_{i=1}^4 u_i \alpha_i$ , where  $(\alpha_1, \dots, \alpha_4)$  is a basis of  $\mathcal{O}$
  - 5: **until**  $\gcd(\text{nrd}(\alpha), N) = 1$
  - 6: Return  $I := \mathcal{O}\gamma\alpha + N\mathcal{O}$
- 

## 4 Detailed Description of SQIsign2D

We now present a full description of the SQIsign2D protocol. We start by describing the  $\Sigma$ -protocol underlying SQIsign2D, and then we present the variant of the Fiat-Shamir transform [21] that we rely on to obtain a digital signature protocol.

The protocol uses a field characteristic of the form  $p = c \cdot 2^e - 1$ , where  $c$  is a small cofactor and  $\log p \approx 2\lambda$ . This is already an improvement over existing SQIsign protocols: since such primes are abundant, it is significantly easier to find parameters, especially at higher security levels, for SQIsign2D than for SQIsign. Compared to SQIsignHD, which uses Montgomery-friendly primes  $p = c \cdot 2^e \cdot 3^f - 1$ , SQIsign2D primes offer even better opportunities for low-level optimisations, as discussed in Sect. 6.

### 4.1 The $\Sigma$ -Protocol

**Key Generation.** During key generation, we sample a random left ideal  $I_{\text{sk}}$  of  $\mathcal{O}_0$  of norm  $N_{\text{sk}}$  via RandomFixedNormIdeal (Algorithm 3), where  $N_{\text{sk}}$  is an odd integer of size  $4\lambda$ . The ideal  $I_{\text{sk}}$  corresponds to the isogeny  $\varphi_{\text{sk}}: E_0 \rightarrow E_{\text{pk}}$  connecting  $E_0$  to the public key  $E_{\text{pk}}$ . To be more precise, we compute  $E_{\text{pk}}$  via IdealTolsogeny.

From a mathematical perspective, the ideal  $I_{\text{sk}}$  provides enough information to describe the secret isogeny  $\varphi_{\text{sk}}$ . However, in order to speed up the response algorithm, we perform additional computations that are stored as internal optimisations – we colour these lines to describe such computations. These internal optimisations are required to obtain a faster translation from the challenge to its corresponding ideal; we will formalise what we mean with “its corresponding ideal” in the paragraph “Response” below.

The gist of these optimisations is to evaluate a basis  $\{\beta_1, \beta_2, \beta_3, \beta_4\}$  of the right order  $\mathcal{O}_{\text{pk}}$  of  $I_{\text{sk}}$  at the  $2^e$ -torsion of  $E_{\text{pk}}$ . This is achieved via [11, Algorithm 9]. The key-generation procedure is formalised in Algorithm 4.

**Commitment.** The commitment phase is similar to the key-generation computations: as explained above, we first sample a random left ideal  $I_{\text{com}}$  of  $\mathcal{O}_0$  of norm  $N_{\text{com}} = \ell_{\text{com}}^n$ , for some  $n > 0$ . In particular, we require  $\ell_{\text{com}} > 2^{e_{\text{rsp}}}$ , where  $2^{e_{\text{rsp}}}$  denotes the largest possible degree of the response isogeny. This condition implies that we can compute the pushforward of any left ideal  $I$  of  $\mathcal{O}_0$  of

---

**Algorithm 4.** Key Generation

---

**Output:** The public key  $\text{pk} = E_{\text{pk}}$  and the secret key  $\text{sk} = I_{\text{sk}}$ .

- 1:  $I_{\text{sk}} \leftarrow \text{RandomFixedNormIdeal}(N_{\text{sk}})$
  - 2:  $\varphi_{\text{sk}}|_{2^e}, E_{\text{pk}} \leftarrow \text{IdealTolsogeny}(I_{\text{sk}}, P_0, Q_0)$ .
  - 3: Compute a deterministic basis  $(P_{\text{pk}}, Q_{\text{pk}})$  of  $E_{\text{pk}}[2^e]$ .
  - 4: Compute a basis  $B = (\beta_1, \beta_2, \beta_3, \beta_4)$  of the right order  $\mathcal{O}_{\text{pk}}$  of  $I_{\text{pk}}$ .
  - 5: Compute the basis  $(\beta_1, \beta_2, \beta_3, \tilde{\beta}_4)$  of  $\text{End}(E_{\text{pk}})$  corresponding to  $B$ .  
(▷) [11, Algorithm 9]
  - 6: Compute  $\mathcal{B} = \left\{ \tilde{\beta}_i(P_{\text{pk}}), \tilde{\beta}_i(Q_{\text{pk}}) \right\}_{i=1, \dots, 4}$ .
  - 7: **return**  $\text{pk} := E_{\text{pk}}$  and  $\text{sk} := (I_{\text{sk}}, \mathcal{B})$
- 

norm  $< 2^{e_{\text{rsp}}}$  under  $I_{\text{com}}$ , which is a necessary step in the response computation (see Algorithm 6, Line 9).

One of the outputs of the Commitment algorithm is the curve  $E_{\text{com}}$  obtained by applying `IdealTolsogeny` on  $I_{\text{com}}$ . Additionally, the algorithm outputs the internal state  $I_{\text{com}}$ . Similarly to what has been said above, the ideal  $I_{\text{com}}$  provides enough information to compute the corresponding isogeny  $\varphi_{\text{com}}: E_0 \rightarrow E_{\text{com}}$ . However, as an internal optimisation, we also extract and store the isogeny representation  $\varphi_{\text{com}}|_{2^e}$ . We summarise everything in Algorithm 5.

---

**Algorithm 5.** Commitment

---

**Output:** The commitment curve  $E$ , and the corresponding state  $I$ ,

- 1:  $I_{\text{com}} \leftarrow \text{RandomFixedNormIdeal}(N_{\text{com}})$ .
  - 2:  $\varphi_{\text{com}}|_{2^e}, E_{\text{com}} \leftarrow \text{IdealTolsogeny}(I_{\text{com}}, P_0, Q_0)$ .
  - 3: **return**  $\text{com} := E_{\text{com}}$  and  $\text{st} := (I_{\text{com}}, \varphi_{\text{com}}|_{2^e})$ .
- 

**Challenge.** The challenge consists of a positive integer  $\text{chl} < 2^{e_{\text{chl}}}$ , where  $e_{\text{chl}}$  is a parameter denoting the size of the challenge space. This integer describes the kernel of the challenge isogeny  $\varphi_{\text{chl}}: E_{\text{pk}} \rightarrow E_{\text{chl}}$ ; i.e.  $\ker(\varphi_{\text{chl}}) = \langle P_{\text{pk}} + [\text{chl}]Q_{\text{pk}} \rangle$ .

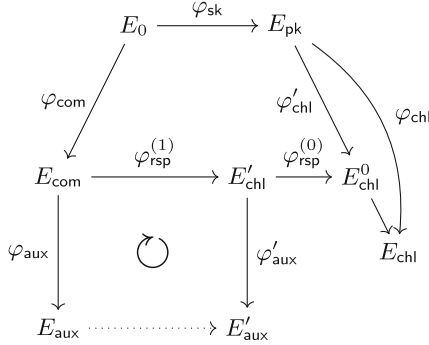
It is worth noting that, although  $\deg(\varphi_{\text{chl}}) = 2^e$ , the challenge space contains only  $2^{e_{\text{chl}}} \ll 2^e$  possible challenges, i.e. we only allow  $2^{e_{\text{chl}}}$  possible kernels. Intuitively, the extra length of  $\varphi_{\text{chl}}$  is needed to deal with the fact that response isogenies may backtrack with  $\varphi_{\text{chl}}$ . This concept is formalised in Theorem 17.

**Response.** The diagram to keep in mind as we explain the response algorithm is the following one (see Fig. 2), where

- $\varphi_{\text{chl}}: E_{\text{pk}} \rightarrow E_{\text{chl}}$  is the isogeny described by the challenge chl;
- $\varphi'_{\text{chl}}: E_{\text{pk}} \rightarrow E_{\text{chl}}^0$  is the portion of  $\varphi_{\text{chl}}$  that does not backtrack with the response isogeny;
- $\varphi_{\text{rsp}}^{(1)}: E_{\text{com}} \rightarrow E'_{\text{chl}}$  is the odd part of the response isogeny;
- $\varphi_{\text{rsp}}^{(0)}: E'_{\text{chl}} \rightarrow E_{\text{chl}}^0$  is the even, non-backtracking part of the response isogeny;



- $\varphi_{\text{aux}}: E_{\text{com}} \rightarrow E_{\text{aux}}$  is the auxiliary isogeny needed to embed the isogeny  $\varphi_{\text{rsp}}^{(1)}$  into a two-dimensional isogeny;
- $\varphi'_{\text{aux}}: E'_{\text{chl}} \rightarrow E'_{\text{aux}}$  is the pushforward of  $\varphi_{\text{aux}}$  under  $\varphi_{\text{rsp}}^{(1)}$ .



**Fig. 2.** Response diagram.

The first step is to compute the ideal  $I_{\text{chl}}$  corresponding to the isogeny  $\varphi_{\text{chl}}: E_{\text{pk}} \rightarrow E_{\text{chl}}$  with kernel  $\langle P_{\text{pk}} + [\text{chl}]Q_{\text{pk}} \rangle$ . This is done via [11, Algorithm 9] using the datum  $\mathcal{B} = \left\{ \tilde{\beta}_i(P_{\text{pk}}), \tilde{\beta}_i(Q_{\text{pk}}) \right\}_{i=1, \dots, 4}$  computed during Key Generation (see Algorithm 4).

The ideal  $I_{\text{chl}}$  is then employed to compute an isogeny  $\varphi_{\text{rsp}}: E_{\text{com}} \rightarrow E_{\text{chl}}$ . To be more precise, the prover first computes an ideal  $I_{\text{rsp}}$ , equivalent to  $\overline{I_{\text{com}}} \cdot I_{\text{sk}} \cdot I_{\text{chl}}$ , which is uniformly distributed among the equivalent ideals of norm  $< 2^{e_{\text{rsp}}}$ . The protocol parameter  $e_{\text{rsp}}$  is chosen such that the existence of  $I_{\text{rsp}}$  (or equivalently a connecting isogeny of degree  $< 2^{e_{\text{rsp}}}$  between  $E_{\text{com}}$  and  $E_{\text{chl}}$ ) is guaranteed, which means that  $2^{e_{\text{rsp}}}$  must be larger than  $2\sqrt{2p}/\pi$ . The norm of  $I_{\text{rsp}}$  must be bounded by  $2^{e_{\text{rsp}}}$  so that we can represent  $\varphi_{\text{rsp}}: E_{\text{com}} \rightarrow E_{\text{chl}}$  via a two-dimensional  $2^{e_{\text{rsp}}}$ -isogeny. In particular, following Kani’s Lemma (Theorem 4), the degree of the one-dimensional isogenies represented by such a two-dimensional isogeny must be odd, but this might not be the case for  $\varphi_{\text{rsp}}$ . We now explain how to deal with the case of even-degree.

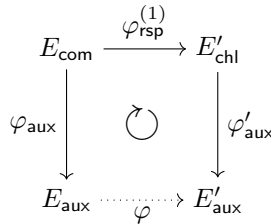
Let us write the norm of  $I_{\text{rsp}}$  as  $\text{nrd}(I_{\text{rsp}}) = q = 2^n q' < 2^{e_{\text{rsp}}}$  for an odd  $q'$ . We can think of  $\varphi_{\text{rsp}}$  as  $\varphi_{\text{rsp}} = \psi \circ \varphi_{\text{rsp}}^{(1)}: E_{\text{com}} \rightarrow E'_{\text{chl}} \rightarrow E_{\text{chl}}$ , where  $\text{deg}(\varphi_{\text{rsp}}^{(1)}) = q'$  and  $\text{deg}(\psi) = 2^n$ . It may happen that  $\ker(\widehat{\psi}) \cap \ker(\widehat{\varphi_{\text{chl}}})$  is not trivial. Let  $n_{\text{bt}}$  be the positive integer such that  $2^{n_{\text{bt}}} = \# \ker(\widehat{\psi}) \cap \ker(\widehat{\varphi_{\text{chl}}})$ . Equivalently,  $n_{\text{bt}}$  is the largest integer such that  $I_{\text{chl}} \cdot \overline{I_{\text{rsp}}} \in 2^{n_{\text{bt}}} \mathcal{O}_{\text{pk}}$ .

Let  $r' := n - n_{\text{bt}}$  and define  $\varphi_{\text{rsp}}^{(0)}: E'_{\text{chl}} \rightarrow E_{\text{chl}}^0$  to be the isogeny with kernel  $\ker(\psi)[2^{r'}]$  – the isogeny  $\varphi_{\text{rsp}}^{(0)}$  coincides with the non-backtrack portion of  $\varphi_{\text{rsp}}$ . Now, let us factor  $I_{\text{rsp}}$  as  $I_{\text{rsp}}^1 \cdot I_{\text{rsp}}^0 \cdot I'$ , where  $\text{nrd}(I_{\text{rsp}}^1) = q'$  and  $\text{nrd}(I_{\text{rsp}}^0) = 2^{r'}$ . The isogenies  $\varphi_{\text{rsp}}^{(1)}$  and  $\varphi_{\text{rsp}}^{(0)}$  correspond to  $I_{\text{rsp}}^1$  and  $I_{\text{rsp}}^0$ , respectively.

Since  $\varphi_{\text{rsp}}^{(1)}$  has odd degree, it can be represented via a  $2^{e_{\text{rsp}}-n}$ -isogeny in dimension 2 by Kani’s Lemma. This requires computing an auxiliary isogeny  $\varphi'_{\text{aux}}: E'_{\text{chl}} \rightarrow E'_{\text{aux}}$  of degree  $2^{e_{\text{rsp}}-n} - q'$ .

As required in Theorem 22, we need the isogeny  $\varphi'_{\text{aux}}: E'_{\text{chl}} \rightarrow E'_{\text{aux}}$  to be uniformly sampled among all the isogenies of degree  $2^{e_{\text{rsp}}-n} - q'$ . Hence, the prover samples a random left ideal  $I''_{\text{aux}}$  of  $\mathcal{O}_0$  of norm  $2^{e_{\text{rsp}}-n} - q'$  and then computes  $I'_{\text{aux}}$  as the pushforward  $I'_{\text{aux}}$  of  $I''_{\text{aux}}$  through  $I_{\text{com}} \cdot I_{\text{rsp}}^{(1)}$ . The prover can then evaluate  $\varphi'_{\text{aux}} \circ \varphi_{\text{rsp}}^{(1)} \circ \varphi_{\text{com}}$  at the  $2^e$ -torsion running IdealTolsogeny on input  $I_{\text{com}} \cdot I_{\text{rsp}}^{(1)} \cdot I'_{\text{aux}}$ . Using the datum  $\varphi_{\text{com}}|_{2^e}$ , the prover has actually access to  $\varphi'_{\text{aux}} \circ \varphi_{\text{rsp}}^{(1)}|_{2^e}$ .

While a representation of  $\varphi'_{\text{aux}} \circ \varphi_{\text{rsp}}^{(1)}$  could act as a valid response, we want the  $\Sigma$ -protocol to be *commitment recoverable*, i.e. it is possible to recompute the commitment curve from a the challenge and corresponding response. This eventually leads to a more compact signature. To achieve such a property, we want the isogeny connecting  $E_{\text{aux}}$  and  $E'_{\text{chl}}$ , passing through  $E_{\text{com}}$ . Thus, the prover has to compute the isogeny  $\varphi_{\text{aux}}: E_0 \rightarrow E_{\text{aux}}$  of degree  $2^{e_{\text{rsp}}-n} - q'$  fitting in the following commutative diagram:



Such an isogeny can be obtained as one of the components of the two-dimensional  $2^{e_{\text{rsp}}-n}$ -isogeny  $\Phi$  with kernel  $\{([q]P, \varphi'_{\text{aux}} \circ \varphi_{\text{rsp}}^{(1)}(P)) \mid P \in E_{\text{com}}[2^{e_{\text{rsp}}-n}]\}$ :

$$\Phi = \begin{pmatrix} \varphi_{\text{rsp}}^{(1)} & -\widehat{\varphi'_{\text{aux}}} \\ \varphi_{\text{aux}} & \widehat{\varphi} \end{pmatrix} : E_{\text{com}} \times E'_{\text{aux}} \rightarrow E'_{\text{chl}} \times E_{\text{aux}}.$$

To complete the response algorithm, we still need to compute the non-backtracking part of the response isogeny. Let  $\varphi_{\text{rsp}}^{(0)}: E'_{\text{chl}} \rightarrow E_{\text{chl}}^0$  be such an isogeny, which indeed corresponds to the ideal  $I_{\text{rsp}}^0$ .

Let  $\varphi'_{\text{chl}}: E_{\text{pk}} \rightarrow E_{\text{chl}}^0$  be the isogeny with kernel  $\langle [2^{n_{\text{bt}}}] (P_{\text{pk}} + [\text{chl}]Q_{\text{pk}}) \rangle$ . In other words,  $\varphi'_{\text{chl}}$  is the portion of  $\varphi_{\text{chl}}$  that does not backtrack with the response isogeny. Even though  $\varphi'_{\text{chl}}$  and  $\varphi_{\text{rsp}}^{(0)}$  map onto the same elliptic curve, the curves obtained after an explicit computation of the two isogenies will only be equal up to isomorphism. Thus, the prover additionally has to compute an explicit isomorphism to let the two curves agree.

The explicit computation of the isomorphism between the codomains of  $\varphi'_{\text{chl}}$  and  $\varphi_{\text{rsp}}^{(0)}$  is required to facilitate the verification. During the verification, the verifier will not compute  $\varphi_{\text{chl}}$  but rather compute its non-backtrack portion, i.e. the verifier will only compute the isogeny with kernel  $\langle [2^{n_{\text{bt}}}] (P_{\text{pk}} + [\text{chl}]Q_{\text{pk}}) \rangle$ .

Let  $\{P_{\text{aux}}, Q_{\text{aux}}\}$  be a deterministic basis of  $E_{\text{aux}}[2^{e_{\text{rsp}}-n_{\text{bt}}}]$  and define

$$P_{\text{chl}} := [2^{e_{\text{rsp}}-n} - q']^{-1} \varphi_{\text{rsp}}^{(0)} \circ \varphi_{\text{rsp}}^{(1)}(P_{\text{aux}}), \quad Q_{\text{chl}} := [2^{e_{\text{rsp}}-n} - q']^{-1} \varphi_{\text{rsp}}^{(0)} \circ \varphi_{\text{rsp}}^{(1)}(Q_{\text{aux}}).$$

The output of the response algorithm consists in  $(E_{\text{aux}}, P_{\text{chl}}, Q_{\text{chl}}, r', n_{\text{bt}})$ . We collect what has been explained in this paragraph in Algorithm 6.

---

**Algorithm 6.** Response
 

---

**Input:** The public key  $E_{\text{pk}}$ , the secret key  $I_{\text{sk}}, \mathcal{B}$ , the commitment  $(E_{\text{com}}, \text{com})$ , the commitment state  $I_{\text{com}}, \varphi_{\text{com}}(P_0), \varphi_{\text{com}}(Q_0)$ , and the challenge  $\text{chl} < 2^{e_{\text{chl}}}$ .

**Output:**  $E_{\text{aux}}, P_{\text{aux}}, Q_{\text{aux}}, r', n_{\text{bt}}$

- 1: Compute a deterministic basis  $(P_{\text{pk}}, Q_{\text{pk}})$  of  $E_{\text{pk}}[2^e]$ .
  - 2: Compute the ideal  $I_{\text{chl}}$  from  $\text{chl}$  and using  $\mathcal{B}$ . (▷) [11, Algorithm 9]  
 $\varphi_{\text{chl}} : E_{\text{pk}} \rightarrow E_{\text{chl}}$  is the isogeny with kernel  $\langle P_{\text{pk}} + [\text{chl}]Q_{\text{pk}} \rangle$ .
  - 3: Set  $J = \overline{I_{\text{com}}} \cdot I_{\text{sk}} \cdot I_{\text{chl}}$ .
  - 4: Compute a uniformly distributed ideal  $I_{\text{rsp}}$  equivalent to  $J$  of norm  $q < 2^{e_{\text{rsp}}}$ .
  - 5: Compute  $n$  such that  $q = q' \cdot 2^n$ , where  $q'$  is odd and  $n_{\text{bt}} < n$  as the largest integer such that  $I_{\text{chl}} \cdot \overline{I_{\text{rsp}}} \in 2^{n_{\text{bt}}} \mathcal{O}_{\text{pk}}$ .  
//  $n_{\text{bt}}$  is the length of the part of the response that backtracks along the challenge isogeny
  - 6:  $r' \leftarrow n - n_{\text{bt}}$ .
  - 7: Factor  $I_{\text{rsp}}$  as  $I_{\text{rsp}}^1 \cdot I_{\text{rsp}}^0 \cdot I'$  where  $\text{nrd}(I_{\text{rsp}}^1) = q'$  and  $\text{nrd}(I_{\text{rsp}}^0) = 2^{r'}$ .  
//  $I_{\text{rsp}}^1$  is the ideal corresponding to the odd part of the response isogeny  $\varphi_{\text{rsp}}^{(1)} : E_{\text{com}} \rightarrow E'_{\text{chl}}$ , and  $I_{\text{rsp}}^0$  is the ideal corresponding to the even part of the response isogeny  $\varphi_{\text{rsp}}^{(0)} : E'_{\text{chl}} \rightarrow E_{\text{chl}}$ .
  - 8:  $I''_{\text{aux}} \leftarrow \text{RandomFixedNormIdeal}(2^{e_{\text{rsp}}-n} - q')$ .
  - 9: Compute  $I'_{\text{aux}}$  as the pushforward of  $I''_{\text{aux}}$  through  $I_{\text{com}} \cdot I_{\text{rsp}}^1$ .  
//  $I'_{\text{aux}}$  is the ideal corresponding to an auxiliary isogeny  $\varphi'_{\text{aux}} : E'_{\text{chl}} \rightarrow E_{\text{aux}}$ .
  - 10:  $\varphi'_{\text{aux}} \circ \varphi_{\text{rsp}}^{(1)} \circ \varphi_{\text{com}} \Big|_{2^e}, E'_{\text{aux}} \leftarrow \text{IdealToIsogeny}(I_{\text{com}} \cdot I_{\text{rsp}}^1 \cdot I'_{\text{aux}})$ .
  - 11:  $P_{\text{com}}^0, Q_{\text{com}}^0 \leftarrow [2^{e-(e_{\text{rsp}}-n)}] \varphi_{\text{com}}(P_0), [2^{e-(e_{\text{rsp}}-n)}] \varphi_{\text{com}}(Q_0)$ .
  - 12:  $P'_{\text{aux}}, Q'_{\text{aux}} \leftarrow [2^{e-(e_{\text{rsp}}-n)}] \varphi'_{\text{aux}} \circ \varphi_{\text{rsp}}^{(1)} \circ \varphi_{\text{com}}(P_0), [2^{e-(e_{\text{rsp}}-n)}] \varphi'_{\text{aux}} \circ \varphi_{\text{rsp}}^{(1)} \circ \varphi_{\text{com}}(Q_0)$ .
  - 13: Compute  $\Phi' : E_{\text{com}} \times E'_{\text{aux}} \rightarrow E'_{\text{chl}} \times E_{\text{aux}}$  with kernel  $\langle ([q']P_{\text{com}}^0, P'_{\text{aux}}), ([q']Q_{\text{com}}^0, Q'_{\text{aux}}) \rangle$
  - 14:  $(\tilde{P}_{\text{chl}}, \tilde{P}_{\text{aux}}) \leftarrow \Phi'(\varphi_{\text{com}}(P_0), 0)$ .
  - 15:  $(\tilde{Q}_{\text{chl}}, \tilde{Q}_{\text{aux}}) \leftarrow \Phi'(\varphi_{\text{com}}(Q_0), 0)$ .
  - 16:  $E_{\text{chl}}^0 \leftarrow E'_{\text{chl}}$ .
  - 17: **if**  $r' > 0$  **then**
  - 18: Compute the isogeny  $\varphi_{\text{rsp}}^0 : E'_{\text{chl}} \rightarrow E_{\text{chl}}^0$  corresponding to  $I_{\text{rsp}}^0$ .
  - 19:  $\tilde{P}_{\text{chl}}, \tilde{Q}_{\text{chl}} \leftarrow \varphi_{\text{rsp}}^0(\tilde{P}_{\text{chl}}), \varphi_{\text{rsp}}^0(\tilde{Q}_{\text{chl}})$ .
  - 20: Compute  $\varphi'_{\text{chl}} : E_{\text{pk}} \rightarrow (E_{\text{chl}}^0)'$  of kernel  $\langle [2^{n_{\text{bt}}}] (P_{\text{pk}} + [\text{chl}]Q_{\text{pk}}) \rangle$ .
  - 21: Compute the isomorphism  $\iota_{\text{chl}} : E_{\text{chl}}^0 \rightarrow (E_{\text{chl}}^0)'$ .
  - 22:  $\tilde{P}_{\text{chl}}, \tilde{Q}_{\text{chl}} \leftarrow \iota_{\text{chl}}(\tilde{P}_{\text{chl}}), \iota_{\text{chl}}(\tilde{Q}_{\text{chl}})$ .
  - 23: Compute a deterministic basis  $(P_{\text{aux}}, Q_{\text{aux}})$  of  $E_{\text{aux}}[2^{e_{\text{rsp}}-n_{\text{bt}}}]$ .
  - 24: Compute  $a, b, c, d \in \mathbb{Z}/2^{e_{\text{rsp}}-n_{\text{bt}}}\mathbb{Z}$  such that  

$$P_{\text{aux}} = [2^{e-e_{\text{rsp}}+n_{\text{bt}}}]([a]\tilde{P}_{\text{aux}} + [b]\tilde{Q}_{\text{aux}}) \quad \text{and} \quad Q_{\text{aux}} = [2^{e-e_{\text{rsp}}+n_{\text{bt}}}]([c]\tilde{P}_{\text{aux}} + [d]\tilde{Q}_{\text{aux}}).$$
  - 25:  $P_{\text{chl}}, Q_{\text{chl}} \leftarrow [2^{e-e_{\text{rsp}}+n_{\text{bt}}}]([a]\tilde{P}_{\text{chl}} + [b]\tilde{Q}_{\text{chl}}), [2^{e-e_{\text{rsp}}+n_{\text{bt}}}]([c]\tilde{P}_{\text{chl}} + [d]\tilde{Q}_{\text{chl}})$
  - 26: **return**  $E_{\text{aux}}, P_{\text{chl}}, Q_{\text{chl}}, r', n_{\text{bt}}$ .
-

**Verification.** On input  $(E_{\text{aux}}, P_{\text{chl}}, Q_{\text{chl}}, r', n_{\text{bt}})$ , the verifier first computes the isogeny  $\varphi_{\text{chl}} : E_0 \rightarrow E_{\text{chl}}$  with kernel  $\langle [2^{n_{\text{bt}}}] (P_{\text{pk}} + [\text{chl}] Q_{\text{pk}}) \rangle$  – this corresponds to the non-backtrack portion of the challenge isogeny as in the previous paragraph. Additionally, they compute  $(P_{\text{aux}}, Q_{\text{aux}})$ , a deterministic basis of  $E_{\text{aux}}[2^{e_{\text{rsp}} - n_{\text{bt}}}]$

If  $r' > 0$ , it means that the prover has chosen a response isogeny having an even, non-backtrack component. In this case,  $[2^{e_{\text{rsp}} - r' - n_{\text{bt}}}] P_{\text{chl}}$  and  $[2^{e_{\text{rsp}} - r' - n_{\text{bt}}}] Q_{\text{chl}}$  are linearly dependent, and  $\langle [2^{e_{\text{rsp}} - r' - n_{\text{bt}}}] P_{\text{chl}}, [2^{e_{\text{rsp}} - r' - n_{\text{bt}}}] Q_{\text{chl}} \rangle$  is the kernel of the dual of the isogeny  $\varphi_{\text{rsp}}^{(0)}$  (Cfr. Fig. 2). The verifier then computes the isogeny  $\varphi : E_{\text{chl}} \rightarrow E'_{\text{chl}}$  with kernel  $\langle [2^{e_{\text{rsp}} - r' - n_{\text{bt}}}] P_{\text{chl}}, [2^{e_{\text{rsp}} - r' - n_{\text{bt}}}] Q_{\text{chl}} \rangle$  and updates  $E_{\text{chl}} \leftarrow E'_{\text{chl}}$ ,  $P_{\text{chl}} \leftarrow \varphi(P_{\text{chl}})$  and  $Q_{\text{chl}} \leftarrow \varphi(Q_{\text{chl}})$ .

From Kani’s Lemma, it follows that the isogeny  $\Phi$  with kernel

$$\langle (P_{\text{chl}}, [2^{r'}] P_{\text{aux}}), (Q_{\text{chl}}, [2^{r'}] Q_{\text{aux}}) \rangle$$

maps  $E'_{\text{chl}} \times E_{\text{aux}}$  onto  $E_{\text{aux}} \times E_{\text{com}}$ . This proves the existence of an isogeny connecting  $E_{\text{com}}$  and  $E'_{\text{chl}}$ . We summarise these steps in Algorithm 7.

*Remark 12 (Technical Remark).* In the concrete instantiation, when computing the isogeny  $\Phi$  with kernel  $\mathcal{K} = \langle (P_{\text{chl}}, [2^{r'}] P_{\text{aux}}), (Q_{\text{chl}}, [2^{r'}] Q_{\text{aux}}) \rangle$ , we use the formulae in [12]. In particular, in order to avoid the computation of extra square roots in the codomain computation, we use the four torsion above  $\mathcal{K}$ . As explained in [11, Theorem 56], this also fixes a symplectic four-torsion basis on the codomain, which in turns defines a theta structure.

In the implementation, we always pick the four-torsion above  $\mathcal{K}$  such that the codomain is of the form  $E'_{\text{aux}} \times E_{\text{com}}$ . Therefore, in Algorithm 7, Line 15, we can restrict ourselves to checking that  $F_2$  is isomorphic to  $E_{\text{com}}$ .

## 4.2 The Signature Protocol

To transform the  $\Sigma$ -protocol in a digital signature, we rely on the Fiat-Shamir transform [21], where the interactive challenge generation is replaced by hashing the commitment, together with the message, to obtain a challenge. However, our protocol differs from a straightforward application of the transform: we rely on the commitment-recoverability property of the underlying  $\Sigma$ -protocol to obtain a smaller signature. Namely, a signature of SQIsign2D consists only of a challenge and the corresponding response. To verify a signature, the verifier recovers the challenge from the signature, checks that the commitment, challenge, and response form a valid transcript for the  $\Sigma$ -protocol, and ensures that the challenge was honestly generated.

For this approach to work, it is necessary that the verifier can extract the commitment from the response. During verification, the verifier first computes the challenge isogeny codomain, and then they obtain the two-dimensional isogeny  $\Phi$  (see Line 12 of Algorithm 7). The codomain of  $\Phi$  is either the product  $E'_{\text{aux}} \times E_{\text{com}}$  or  $E_{\text{com}} \times E'_{\text{aux}}$ . While a priori it is not possible to distinguish between the two cases, we rely on a specific method to compute  $\Phi$ , as explained in Remark 12,

**Algorithm 7.** Verify

---

**Input:** The public key  $E_{\text{pk}}$ , the commitment  $E_{\text{com}}$ , the challenge  $\text{chl}$ , the response  $E_{\text{aux}}, P_{\text{chl}}, Q_{\text{chl}}, r', n_{\text{bt}}$ .

**Output:** true or false.

- 1: Compute a deterministic basis  $(P_{\text{pk}}, Q_{\text{pk}})$  of  $E_{\text{pk}}[2^e]$ .
  - 2: Compute  $\varphi_{\text{chl}} : E_0 \rightarrow E_{\text{chl}}$  with kernel  $\langle [2^{n_{\text{bt}}}] (P_{\text{pk}} + [\text{chl}] Q_{\text{pk}}) \rangle$ .
  - 3: Compute a deterministic basis  $(P_{\text{aux}}, Q_{\text{aux}})$  of  $E_{\text{aux}}[2^{e_{\text{rsp}} - n_{\text{bt}}}]$ .
  - 4: **if**  $r' > 0$  **then**
  - 5:     **if**  $[2^{e_{\text{rsp}} - n_{\text{bt}} - 1}] Q_{\text{chl}} \neq 0$  **then**
  - 6:          $R \leftarrow [2^{e_{\text{rsp}} - n_{\text{bt}} - r'}] Q_{\text{chl}}$
  - 7:     **else**
  - 8:          $R \leftarrow [2^{e_{\text{rsp}} - n_{\text{bt}} - r'}] P_{\text{chl}}$
  - 9:     Compute  $\varphi : E_{\text{chl}} \rightarrow E'_{\text{chl}}$  of kernel  $\langle R \rangle$ .
  - 10:      $E_{\text{chl}} \leftarrow E'_{\text{chl}}$ .
  - 11:      $P_{\text{chl}}, Q_{\text{chl}} \leftarrow \varphi(P_{\text{chl}}), \varphi(Q_{\text{chl}})$ .
  - 12: Compute  $\Phi : E_{\text{chl}} \times E_{\text{aux}} \rightarrow F_1 \times F_2$  with kernel  $\langle (P_{\text{chl}}, [2^{r'}] P_{\text{aux}}), (Q_{\text{chl}}, [2^{r'}] Q_{\text{aux}}) \rangle$ .
  - 13: **if** the computation of  $\Phi$  fails **then**
  - 14:     **return** false
  - 15: **return**  $F_2 \cong E_{\text{com}}$
- 

that guarantees that the codomain is  $E'_{\text{aux}} \times E_{\text{com}}$ . Hence, the verifier can extract the commitment curve  $E_{\text{com}}$  from the codomain of  $\Phi$  and check the challenge has been honestly generated, i.e. as the output of the hashing of  $E_{\text{com}}$  and the message to be signed.

## 5 Security Analysis

In this section, we prove that the identification protocol (and thereby the signature scheme obtained by the Fiat–Shamir transform) is secure: it is knowledge-sound and honest-verifier zero-knowledge.

First, note that the key recovery problem for our construction is simply the standard *Supersingular Endomorphism Ring* problem, a foundational problem of isogeny-based cryptography.

*Problem 13 (Supersingular Endomorphism Ring problem).* Given a supersingular elliptic curve  $E/\mathbb{F}_{p^2}$ , find four endomorphisms (in efficient representation) which generate the ring  $\text{End}(E)$ .

The fastest known algorithms for this problem have classical complexity in  $\tilde{O}(p^{1/2})$  [16] (see also [35, Theorem 8.8]). The only known quantum speed-up is using Grover’s algorithm [6, 22], for a quantum complexity in  $\tilde{O}(p^{1/4})$ .

We prove in Theorem 17 that if  $e_{\text{chl}} + e_{\text{rsp}} \leq e$ , the protocol has the 2-special soundness property for the language

$$\{(E_{\text{pk}}, \alpha) \mid \alpha \in \text{End}(E_{\text{pk}}) \setminus \mathbb{Z} \text{ in efficient representation}\}.$$

This language corresponds to the *Supersingular One Endomorphism* problem.

*Problem 14 (Supersingular One Endomorphism problem).* Given a supersingular elliptic curve  $E/\mathbb{F}_{p^2}$ , find a non-scalar endomorphism  $\alpha \in \text{End}(E) \setminus \mathbb{Z}$  (in efficient representation).

This One Endomorphism problem is equivalent to the Endomorphism Ring problem [35], i.e., to the key recovery problem for our construction.

Then, we prove in Theorem 22 that if  $N_{\text{com}} \geq 2^{4\lambda}$  and  $2^{e_{\text{rsp}}} \geq 2\sqrt{2p}/\pi$ , then the protocol is statistically honest-verifier zero-knowledge, in a model where the simulator can sample random large-degree isogenies from a given curve (in the classical model, this can only be done efficiently for smooth degree). This model, discussed in Sect. 5.2, is similar to the security model of SQIsignHD [11].

**Impact on Parameter Selection.** In summary, for a security level ensuring  $\lambda$  bits of classical security, one needs to choose a prime  $p = \Theta(2^{2\lambda})$ . To ensure soundness, one needs  $e_{\text{chl}} + e_{\text{rsp}} \leq e$  (recall that  $p \approx 2^e$ , so  $e \approx 2\lambda$ ). To ensure the statistical honest-verifier zero-knowledge property, one needs  $N_{\text{com}} \geq 2^{4\lambda}$  and  $2^{e_{\text{rsp}}} \geq 2\sqrt{2p}/\pi$ .

### 5.1 Knowledge Soundness

**Lemma 15.** *Given a commitment  $E_{\text{com}}$ , a challenge  $\text{chl} < 2^{e_{\text{chl}}}$  (generating the challenge isogeny  $\varphi_{\text{chl}}: E_{\text{pk}} \rightarrow E_{\text{chl}}$ ), and a response  $(E_{\text{aux}}, P_{\text{chl}}, Q_{\text{chl}}, r', n_{\text{bt}})$  passing verification, one can extract in polynomial time an efficient representation of an isogeny  $\tilde{\sigma}: E_{\text{com}} \rightarrow E_{\text{chl}}$  of degree at most  $2^{e_{\text{rsp}}}$ .*

*Proof.* Write  $\psi: E_{\text{chl}}^0 \rightarrow E_{\text{chl}}$  for the last  $n_{\text{bt}}$  steps of the challenge isogeny. Let  $n = r' + n_{\text{bt}}$ . A successful verification ensures that one can extract a  $2^{r'}$ -isogeny

$$\tilde{\varphi}^{(0)}: \tilde{E}'_{\text{chl}} \rightarrow E_{\text{chl}}^0,$$

(for some curve  $\tilde{E}'_{\text{chl}}$ ) and an  $2^{e_{\text{rsp}}-n}$ -isogeny

$$\Phi: \tilde{E}'_{\text{chl}} \times E_{\text{aux}} \rightarrow E_{\text{com}} \times \tilde{E}'_{\text{aux}},$$

(for some curve  $\tilde{E}'_{\text{aux}}$ ), in efficient representation. Composing  $\Phi$  with the inclusion  $E'_{\text{chl}} \rightarrow E'_{\text{chl}} \times E_{\text{aux}}$  and the projection  $E_{\text{com}} \times E'_{\text{aux}} \rightarrow E_{\text{com}}$ , and taking the dual, we obtain an isogeny  $\tilde{\varphi}^{(1)}: E_{\text{com}} \rightarrow E'_{\text{chl}}$  of degree at most  $2^{e_{\text{rsp}}-r'}$ . Let  $\tilde{\sigma} = \psi \circ \tilde{\varphi}^{(0)} \circ \tilde{\varphi}^{(1)}: E_{\text{com}} \rightarrow E_{\text{chl}}$ . It has degree at most

$$\deg(\psi) \deg(\tilde{\varphi}^{(0)}) \deg(\tilde{\varphi}^{(1)}) \leq 2^{n_{\text{bt}}} \cdot 2^{e_{\text{rsp}}-n} \cdot 2^{r'} = 2^{e_{\text{rsp}}},$$

which proves the lemma. □

**Lemma 16.** *Let  $\varphi_{\text{chl}}: E_{\text{pk}} \rightarrow E_{\text{chl}}$  and  $\varphi'_{\text{chl}}: E_{\text{pk}} \rightarrow E'_{\text{chl}}$  be two distinct challenges from the same public curve  $E_{\text{pk}}$ . Then, the largest integer dividing  $\varphi'_{\text{chl}} \circ \hat{\varphi}_{\text{chl}} \in \text{Hom}(E_{\text{chl}}, E'_{\text{chl}})$  is smaller than  $2^{e_{\text{chl}}}$ .*

*Proof.* Recall that the challenge isogeny  $\varphi_{\text{chl}}$  is defined by the kernel  $\langle K(\text{chl}) \rangle$  with

$$K(\text{chl}) = P_{\text{pk}} + [\text{chl}]Q_{\text{pk}}$$

where  $0 \leq \text{chl} < 2^{e_{\text{chl}}}$ , and  $\langle P_{\text{pk}}, Q_{\text{pk}} \rangle = E_{\text{pk}}[2^e]$ . The second challenge isogeny  $\varphi'_{\text{chl}}$  is defined similarly by its kernel generator  $K(\text{chl}') = P_{\text{pk}} + [\text{chl}']Q_{\text{pk}}$ , for some  $\text{chl} \neq \text{chl}'$ . Since  $\varphi_{\text{chl}}$  and  $\varphi'_{\text{chl}}$  are cyclic, by [11, Lemma 37] there exists three cyclic isogenies  $\varphi_0 : E_{\text{pk}} \rightarrow E$ ,  $\varphi_1 : E \rightarrow E_{\text{chl}}$  and  $\varphi'_1 : E \rightarrow E'_{\text{chl}}$  such that  $\varphi_{\text{chl}} = \varphi_1 \circ \varphi_0$ ,  $\varphi'_{\text{chl}} = \varphi'_1 \circ \varphi_0$  and  $\varphi'_1 \circ \hat{\varphi}_1$  is cyclic. We call  $\varphi_0$  the *greatest cyclic factor* of  $\varphi_{\text{chl}}$  and  $\varphi'_{\text{chl}}$ . It has kernel  $\ker(\varphi_0) = \ker(\varphi_{\text{chl}}) \cap \ker(\varphi'_{\text{chl}})$ . Since  $\varphi'_{\text{chl}} \circ \hat{\varphi}_{\text{chl}} = [\deg(\varphi_0)]\varphi'_1 \circ \hat{\varphi}_1$ , we see that  $\deg(\varphi_0)$  is the largest integer dividing  $\varphi'_{\text{chl}} \circ \hat{\varphi}_{\text{chl}}$  in  $\text{Hom}(E_{\text{chl}}, E'_{\text{chl}})$ , so we only have to prove that  $\deg(\varphi_0) < 2^{e_{\text{chl}}}$ .

Let  $R \in E_{\text{pk}}$  be a generator of  $\ker(\varphi_0)$ . Then,  $R = [a]K(\text{chl}) = [b]K(\text{chl}')$  for some  $a, b \in \llbracket 0; 2^e - 1 \rrbracket$ , i.e.,

$$[a - b]P_{\text{pk}} + [a \cdot \text{chl} - b \cdot \text{chl}']Q_{\text{pk}} = 0.$$

Since  $(P_{\text{pk}}, Q_{\text{pk}})$  is a basis of  $E_{\text{pk}}[2^e]$ , it follows that  $a - b \equiv 0 \pmod{2^e}$  so  $a = b$  and  $a(\text{chl} - \text{chl}') \equiv 0 \pmod{2^e}$ . Since  $0 \leq \text{chl} \neq \text{chl}' < 2^{e_{\text{chl}}}$ , it follows that  $2^{e - e_{\text{chl}} + 1} | a$ , so that  $R \in E_{\text{pk}}[2^{e_{\text{chl}} - 1}]$  and  $\deg(\varphi_0) \leq 2^{e_{\text{chl}} - 1}$ . This completes the proof.  $\square$

**Theorem 17.** *If  $e_{\text{chl}} + e_{\text{rsp}} \leq e$ , then the identification protocol has 2-special soundness for the language*

$$\{(E_{\text{pk}}, \alpha) \mid \alpha \in \text{End}(E_{\text{pk}}) \setminus \mathbb{Z} \text{ in efficient representation}\}.$$

*Proof.* Consider two accepting transcripts with the same commitment curve  $E_{\text{com}}$  but challenge isogenies  $\varphi_{\text{chl}} : E_{\text{pk}} \rightarrow E_{\text{chl}}$  and  $\varphi'_{\text{chl}} : E_{\text{pk}} \rightarrow E'_{\text{chl}}$  with distinct kernels. From Lemma 15, we can extract an efficient representation of isogenies  $\sigma : E_{\text{com}} \rightarrow E_{\text{chl}}$  and  $\sigma' : E_{\text{com}} \rightarrow E'_{\text{chl}}$ , each of degree at most  $2^{e_{\text{rsp}}}$ .

Suppose by contradiction that  $\alpha = [m]$  for some  $m \in \mathbb{Z}$ . We deduce

$$[m] \circ \varphi'_{\text{chl}} \circ \hat{\varphi}_{\text{chl}} = [\deg(\varphi_{\text{chl}}) \deg(\varphi'_{\text{chl}})] \circ \sigma' \circ \hat{\sigma}. \tag{2}$$

Write  $\varphi'_{\text{chl}} \circ \hat{\varphi}_{\text{chl}} = [2^a] \circ \psi$  and  $\sigma' \circ \hat{\sigma} = [d] \circ \nu$  where  $\psi$  and  $\nu$  have cyclic kernel. We deduce from Eq. (2) that  $2^a m = d \deg(\varphi_{\text{chl}}) \deg(\varphi'_{\text{chl}})$  is the largest integer dividing either side of the equality, and  $\psi = \nu$  is the cyclic part of either side.

On one hand, we have  $\deg(\nu) \leq \deg(\sigma) \deg(\sigma') \leq 2^{2e_{\text{rsp}}}$ . On the other hand, Lemma 16 implies

$$\deg(\psi) = \frac{\varphi'_{\text{chl}} \circ \hat{\varphi}_{\text{chl}}}{2^{2a}} > 2^{2(e - e_{\text{chl}})} \geq 2^{2e_{\text{rsp}}}.$$

This contradicts the equality  $\psi = \nu$ .  $\square$

### 5.2 Zero-Knowledge Property

In this section, we prove that the identification protocol is honest-verifier zero-knowledge. Let us first prove that the commitment curve is indistinguishable from a uniformly random curve.

**Lemma 18.** *If  $N_{\text{com}} \geq 2^{4\lambda}$ , then an honestly generated commitment curve  $E_{\text{com}}$  is at statistical distance  $\tilde{O}(2^{-\lambda})$  from a uniformly random supersingular elliptic curve.*

*Proof.* It follows from [11, Proposition 29] with  $\varepsilon = 1$  and  $p = \Theta(2^{2\lambda})$ . □

To prove that the protocol has the zero-knowledge property, we prove that there exists a simulator producing transcripts indistinguishable from an honest run of the protocol. Like in SQIsignHD [11], the simulator runs in polynomial time if it has access to an oracle producing random isogenies. This “random isogeny” oracle comes in two variants: the UTO and the FIDIO.

**Definition 19.** *A uniform target oracle (UTO) is an oracle taking as input a supersingular elliptic curve  $E$  defined over  $\mathbb{F}_{p^2}$  and an integer  $N \geq 2\sqrt{2p}/\pi$ , and outputs a random isogeny  $\varphi : E \rightarrow E'$  (in efficient representation) such that:*

1. *The distribution of  $E'$  is uniform among all the supersingular elliptic curves.*
2. *The conditional distribution of  $\varphi$  given  $E'$  is uniform among isogenies  $E \rightarrow E'$  of degree smaller or equal to  $N$ .*

*Remark 20.* The condition  $N \geq 2\sqrt{2p}/\pi$  ensures such an oracle exists: for any pair  $(E_1, E_2)$ , the collection of isogenies  $E_1 \rightarrow E_2$  of degree smaller than  $N$  is non-empty (Minkowski’s bound for the lattice  $\text{Hom}(E_1, E_2)$ ).

**Definition 21.** *A fixed degree isogeny oracle (FIDIO) is an oracle taking as input a supersingular elliptic curve  $E$  defined over  $\mathbb{F}_{p^2}$  and an integer  $N$ , and outputs a uniformly random isogeny  $\varphi : E \rightarrow E'$  (in efficient representation) with domain  $E$  and degree  $N$ .*

**Theorem 22.** *If  $2^{e_{\text{rsp}}} \geq 2\sqrt{2p}/\pi$  and  $N_{\text{com}} \geq 2^{4\lambda}$ , then the identification protocol is statistically honest-verifier zero-knowledge in the UTO and FIDIO model. In other words, there exists a polynomial time simulator  $\mathcal{S}$  with access to a UTO and a FIDIO that produces random transcripts which are statistically indistinguishable from honest transcripts.*

*Proof.* The simulator proceeds as follows:

1. Generate an isogeny  $\varphi_{\text{chl}} : E_{\text{pk}} \rightarrow E_{\text{chl}}$  according to the honest challenge distribution.
2. Call the UTO on input  $(E_{\text{chl}}, 2^{e_{\text{rsp}}})$ , resulting in the isogeny  $\hat{\varphi}_{\text{rsp}} : E_{\text{chl}} \rightarrow E_{\text{com}}$ .
3. Decompose  $\varphi_{\text{rsp}} = \psi \circ \varphi_{\text{rsp}}^{(1)}$  with  $q' = \deg(\varphi_{\text{rsp}}^{(1)})$  odd and  $\deg(\psi) = 2^n$  a power of two. Let  $2^{n_{\text{bt}}} = \#(\ker(\hat{\psi}) \cap \ker(\hat{\varphi}_{\text{chl}}))$ . Let  $r' = n - n_{\text{bt}}$ .
4. Call the FIDIO on input  $(E_{\text{com}}, 2^{e_{\text{rsp}} - r'} - q')$ , resulting in the isogeny  $\varphi_{\text{aux}} : E_{\text{com}} \rightarrow E_{\text{aux}}$ .

From the properties of the UTO and FIDIO, the above procedure is equivalent to the following one:



1. Generate a uniformly random supersingular curve  $E_{\text{com}}$
2. Generate an isogeny  $\varphi_{\text{chl}} : E_{\text{pk}} \rightarrow E_{\text{chl}}$  according to the honest challenge distribution.
3. Generate a uniformly random isogeny  $\varphi_{\text{rsp}}$  from  $E_{\text{com}}$  to  $E_{\text{chl}}$ , of degree at most  $2^{e_{\text{rsp}}}$ .
4. Decompose  $\varphi_{\text{rsp}} = \psi \circ \varphi_{\text{rsp}}^{(1)}$  with  $q' = \deg(\varphi_{\text{rsp}}^{(1)})$  odd and  $\deg(\psi) = 2^n$  a power of two. Let  $2^{n_{\text{bt}}} = \#(\ker(\hat{\psi}) \cap \ker(\hat{\varphi}_{\text{chl}}))$ . Let  $r' = n - n_{\text{bt}}$ .
5. Generate a uniformly random isogeny  $\beta$  from  $E_{\text{com}}$  and of degree  $2^{e_{\text{rsp}} - r'} - q'$ .

This is precisely the order in which an honest run of the protocol proceeds. The distribution for the first step matches the honest run by Lemma 18. The distributions of following steps match the honest ones by construction.  $\square$

**On the UTO and FIDIO Oracles.** Let us first argue that the UTO is essentially redundant: given a FIDIO, one can implement an oracle that is computationally indistinguishable from a UTO, at least when the bound  $N$  is sufficiently large. We proceed in two steps:

1. First, we use the FIDIO to build an oracle which outputs a uniform isogeny  $\sigma$  from  $E$  with  $\deg(\sigma) \leq N$ . In other words, one can turn a FIDIO into a RADIO, following the terminology of [11].
2. Second, we argue that this distribution (the output of a RADIO) is indistinguishable from the output of a UTO.

Recall the definition of a RADIO.

**Definition 23** ([11, Definition 41]). *A random any-degree isogeny oracle (RADIO) is an oracle taking as input a supersingular elliptic curve  $E$  defined over  $\mathbb{F}_{p^2}$  and an integer  $N$ , and outputs a uniformly random isogeny  $\varphi : E \rightarrow E'$  (in efficient representation) with domain  $E$  and degree at most  $N$ .*

Let us first explain how one can turn a FIDIO into a RADIO. Let  $f_N$  be the probability distribution of the degree of the output of a RADIO: for any integer  $q$ , let  $f_N(q)$  be the probability that the degree of the output of a RADIO on input  $(E, N)$  is equal to  $q$ . Note that conditional on the degree of the output being  $q$ , the FIDIO and the RADIO follow the same distribution: uniform among isogenies with domain  $E$  and degree  $q$ . Therefore, to simulate a RADIO, we can proceed as follows: on input  $(E, N)$ ,

1. sample an integer  $q$  following the distribution  $f_N$ ;
2. call the FIDIO on input  $(E, q)$ , and return the output.

To sample from the distribution  $f_N$ , observe that the value  $f_N(q) = \tilde{\Theta}(q/N^2)$  can be computed efficiently if the factorisation of  $q$  is known. Therefore, we can do rejection sampling by sampling uniformly random integers in  $[1, N]$  together with their factorisation (see [1]).

We proceed as follows: sample a random degree  $q \leq N$ , then call the FIDIO to sample a uniform isogeny of degree  $q$  from  $E$ . The only difficulty is to sample

$q \leq N$  with the same distribution as the degree of a UTO-output (it is not the uniform distribution). Given the prime factorisation  $q = \prod_i \ell_i^{e_i}$ , there are  $\prod_i \ell_i^{e_i}$ .

Now that we can turn a FIDIO into a RADIO, it remains to argue that a RADIO is indistinguishable from a UTO. For  $N$  large enough, it is indeed statistically indistinguishable: conditionally on the target curve, the two distributions are identical, and it is proven in [11, Theorem 42] that when  $N = \Theta(p^{1+\varepsilon})$  for  $\varepsilon \in (0, 2]$ , the distribution on the target curves are at statistical distance  $O(p^{-\varepsilon/2})$ . Therefore, when  $N = \Theta(p^{1+\varepsilon})$ , the RADIO and the UTO are at statistical distance  $O(p^{-\varepsilon/2})$ . The bound  $N = O(p^{1/2})$  used in the protocol is not large enough for this theorem to apply, but we expect the distributions to remain computationally indistinguishable.

The conclusion of the above discussion is that in Theorem 22, the UTO is heuristically redundant. In other words, there is a (heuristic) simulator in the FIDIO model. It remains to argue that this FIDIO does not hurt the security assumption: access to a FIDIO does not help with solving the endomorphism ring problem. We refer to the analogous discussion about the security of SQIsignHD in [11, Section 5.3]. In essence, all a FIDIO does is compute a random walk from a source curve. We already know how to compute random walks of smooth degree (by taking a sequence of random isogeny steps of small prime degree), and a FIDIO extends this capability to random walks with potentially large prime steps.

### 5.3 Security of the Signature Protocol

In the previous sections, we have shown that the SQIsign2D  $\Sigma$ -protocol is 2-special sound, under the assumed hardness of Problem 13, and zero-knowledge in the UTO and FIDIO model. Hence, a direct application of the Fiat–Shamir transform [21] yields a digital signature that is EUF-CMA secure in the random oracle model (ROM) [36], under the hardness of Problem 13 when the attacker has access to the UTO and FIDIO.

However, the signature protocol whose security is proved in [36] includes commitments in the signature. As explained in Sect. 4.2, we replace the commitment in the signature with the challenge (by relying on the commitment-recoverability property of the  $\Sigma$ -protocol) to reduce the signature size. To show the security equivalence of the two approaches, we rely on [2, Theorem 2], which requires the commitment-recovering algorithm to be correct and sound. Given a transcript  $(\text{com}, \text{chl}, \text{rsp})$ , correctness requires the commitment-recovering algorithm to always produce  $\text{com}$  given  $\text{chl}$  and  $\text{rsp}$ , and it follows from Remark 12. Soundness, in this context, means that it is computationally hard to find a pair of challenge and response  $(\text{chl}, \text{rsp})$  for which the commitment-recovering algorithm produces a commitment  $\text{com}$  such that  $(\text{com}, \text{chl}, \text{rsp})$  is *not* a valid transcript. In our case, the commitment-recovering algorithm is perfectly sound (i.e. soundness holds even against unbounded adversaries): the curve produced by the commitment-recovering algorithm introduced in Sect. 4.2 is always the codomain of an isogeny, efficiently represented in the response, starting from

$E_{\text{chl}}$ , and the curve  $E_{\text{com}}$  does not need to satisfy any additional requirement to be a valid commitment; thus, the resulting transcript is always valid.

This shows that the SQIsign2D signature protocol is EUF-CMA secure in the ROM, assuming the hardness of Problem 13 when the attacker has also access to the UTO and FIDIO.

## 6 Instantiation and Experimental Results

We selected parameters for the scheme described in Sect. 4 matching NIST post-quantum security levels I, III and V, and implemented them in C building upon the [SQIsign reference implementation](#). We now give details on our implementation and compare its performance to the other variants of SQIsign.

### 6.1 Parameter Choices and and Signature Size

*Choice of the Primes.* As mentioned in Sect. 5, the best attacks against the Supersingular Endomorphism Ring problem have classical complexity  $\tilde{O}(p^{1/2})$  and quantum complexity  $\tilde{O}(p^{1/4})$ , where  $p$  is the characteristic of the base field. These are also the best known attacks against SQIsign (see [9, Chapter 9]) and SQIsign2D. Our security reduction, although not tight and formulated in the UTO/FIDIO model, further justifies using these complexities to set parameters.

To reach NIST’s security levels I, III and V, we thus look for primes of roughly 256, 384 and 512 bits respectively. For maximum efficiency, we selected primes such that  $2p$  fits in 4, 6 and 8 64-bits words. The final requirement is that  $p + 1 = c \cdot 2^e$  with  $c$  as small as possible; it is also desirable that  $c$  has small Hamming weight. Our final choices are listed in Table 2.

**Table 2.** Chosen parameters for SQIsign2D. Sizes in bytes.

	NIST I	NIST III	NIST V
Prime	$5 \cdot 2^{248} - 1$	$65 \cdot 2^{376} - 1$	$27 \cdot 2^{500} - 1$
Public-key size	66	98	130
Signature size	148	222	294

*Signature Encoding and Sizes.* The resulting public key and signature sizes are reported in Table 2. We detail below how these numbers are computed.

As for other SQIsign variants, there are various possibilities to decrease the signature size at the expense of slower verification and signing. For our implementation, we prioritised verification speed over signature size, and thus chose to not use the most advanced compression tricks. As we mentioned already (see Sect. 4.2), our scheme is commitment recoverable which means that we do not need to include the commitment curve in the signature. This requires a little more work for the signer, but it makes close to no difference for the verification.

Outside of this, the only other real compression we use is to represent the basis  $P_{\text{chl}}, Q_{\text{chl}}$  as four elements in  $[0, 2^{e_{\text{rsp}}}]$  (that are the coefficients of  $P_{\text{chl}}, Q_{\text{chl}}$  in a canonical basis of  $E_{\text{chl}}$ ). For a given level security of  $\lambda$ , we have  $\log p \approx 2\lambda$  and  $e_{\text{rsp}} \approx \lambda$ , so this compression allows us to decrease the size of the basis representation from  $8\lambda$  (since each point is represented as one element in  $\mathbb{F}_{p^2}$ ) to  $4\lambda$ . This requires the additional computation a canonical basis of  $E_{\text{chl}}$ . In general, this is not cheap to compute, but we can abuse tricks specialised for the generation of bases of  $E[2^k]$  such as the entangled torsion basis from [43, Algorithm 3.1] or the modification described in [41, Section 5.1].

We can further reduce the cost of the basis generation for the verifier by including hints at the very reasonable cost of increasing the signature size by two bytes. The idea of hints to speed-up basis generation was first introduced as part of the compression procedure in the original SQIsign paper. Using the specialised algorithms [41, 43] boils down to selecting  $x$ -coordinates with chosen Legendre symbols and checking whether the chosen  $x$  is a valid  $x$ -coordinate for a point on the curve.

In this context, the hints can be either indices of tables of “good”  $x$ -coordinates, or some integer  $h$  such that  $x = i + h \in \mathbb{F}_{p^2}$  are values with the correct Legendre symbol properties and points on the curve.<sup>1</sup> Moreover, it does not cost anything to the signer to include these hints. In our experiments, the value of the hints never went over 50, thus we conjecture that for the sizes considered for our scheme, the hints for a basis can fit in two bytes with overwhelming probability.

In our scheme, we use hints for the deterministic basis generation required by the verification: one for  $E_{\text{pk}}$  and one for  $E_{\text{chl}}$ . Thus, this increases the size of the public key by two bytes and the size of the signature by two bytes.

In the end, the size of the public key is  $4\lambda + 16$  bits, and the size of the signature is  $9\lambda + 16 + 2 \log_2(2\lambda)$  bits ( $\lambda$  for the scalar chl,  $4\lambda$  for  $E_{\text{aux}}$ ,  $4\lambda + 16$  for  $P_{\text{chl}}, Q_{\text{chl}}$  and  $2 \log_2(2\lambda)$  for  $r'$  and  $n_{\text{bt}}$ ).

## 6.2 Implementation Choices and Optimisations

We implemented SQIsign2D in C by modifying [SQIsign’s reference code](#).<sup>2</sup>

*Multi-precision integers and quaternion algebras* are built on top of the GMP library.<sup>3</sup> The only significant difference with SQIsign is the use of floating point numbers in the LLL algorithm instead of exact rationals.

*Arithmetic modulo  $p$*  has two implementations: one based on the Fiat-Crypto code generator [20] and one optimised implementation using the special form of

<sup>1</sup> In our implementation, we begin sampling coordinates from two tables with twenty values. This gives a  $2^{-20}$  chance of failure, which we recover from by then sampling coordinates of the form  $x = i + h$  as above. Regardless of whether the basis is generated from a look-up or sampling, the cost for verification is the same thanks to the supplied hint.

<sup>2</sup> Our code will be available at <https://github.com/SQIsign/sqisign2d-west-ac24>.

<sup>3</sup> <https://gmplib.org/>.

the primes used, allowing for efficient Montgomery reduction. We give a detail of the design choices of this implementation and future work in [3, Appendix C].

*Elliptic Curves, Pairings, and Isogenies.* Following standard practice, we represent elliptic curves in Montgomery form and use the formulas in [10,37] to evaluate 2-isogenies and 4-isogenies. Compared to SQIsign, we do not use formulas for isogenies of odd degrees, and in particular we do not need the costly  $\sqrt{\text{élu}}$  algorithm [5].

For pairings, we use the biextension/cubical formulas from [40], because these are currently, to the best of our knowledge, the fastest available to compute pairings on Montgomery curves. We note that since we only need to compute pairings between points of order  $2^e$ , we only need to use biextension doublings.

*Two-dimensional abelian varieties* are represented in theta coordinates and their two-dimensional 2-isogenies are evaluated using the formulas in [12]. We use the projective version of their formulas to remove almost all inversions along the isogeny chain.

All other algorithms are either taken from the [implementation of SQIsignHD](#) or have been written from scratch according to the description in Section 3, with minor deviations to allow for several small optimizations, such as commitment recoverability, bases compression, and hints.

**Table 3.** Performance of SQIsign2D on Intel Xeon Gold 6338 (Ice Lake, 2 GHz), using generic finite field arithmetic (Fiat-Crypto), GMP 6.2.1. Turbo-boost disabled. Timings in  $10^6$  cycles.

	Level	SQIsign	SQIsignHD	SQIsign2D
Keygen	I	2,800	190	120
	III	21,300	—	440
	V	91,600	—	1,070
Sign	I	4,600	115	290
	III	39,300	—	1,040
	V	165,000	—	2,490
Verify	I	93	—	25
	III	641	—	98
	V	2,080	—	247

### 6.3 Performance

We ran benchmarks to compare our implementations to the state of the art. All code was compiled on Ubuntu using clang 14, with flags `-march=native -O3`, dynamically linking to the system GMP library (version 6.2.1). Benchmarks were run on an Intel Xeon Gold 6338 (Ice Lake) CPU clocked at 2 GHz with turbo boost disabled. In Table 3 we compare our pure-C implementation to:

- The reference implementation of SQIsign at <https://github.com/SQIsign/the-sqisign>. Because this uses the same modular arithmetic based on Fiat-Crypto, it is a fair comparison for showcasing the higher-level algorithmic improvements of SQIsign2D.
- The implementation of SQIsignHD at <https://github.com/Pierrick-Dartois/SQIsignHD-lib>. This codebase is momentarily lacking a C implementation of the verification, thus we only benchmark key generation and signatures.

For the optimised pure-C implementation we additionally compare to the implementation of SQIsign [15] at <https://github.com/SQIsign/sqisign-ec23>. This has much better assembly optimisations for finite fields and is generally faster than the reference implementation. However, our implementation is the only one to implement all three NIST levels. We additionally implemented the heuristic variant of SQIsign2D described in [3, Appendix B] and included these results under the label SQIsign2D-H. The results are reported in Table 4.

**Table 4.** Performance of SQIsign2D on Intel Xeon Gold 6338 (Ice Lake, 2 GHz), with finite field arithmetic optimised using intrinsics for the Ice Lake architecture, GMP 6.2.1. Turbo-boost disabled. Timings in  $10^6$  cycles.

	Level	SQIsign ( [9] )	SQIsign ( [15] )	SQIsign2D	SQIsign2D-H
Keygen	I	1,700	400	60	58
	III	—	—	170	170
	V	—	—	360	350
Sign	I	2,400	1880	160	100
	III	—	—	460	280
	V	—	—	940	570
Verify	I	39	29	9	9
	III	—	—	29	29
	V	—	—	62	60

## References

1. Bach, E.: How to generate factored random numbers. *SIAM Journal on Computing* **17**(2), 179–193 (1988). <https://doi.org/10.1137/0217012>
2. Backendal, M., Bellare, M., Sorrell, J., Sun, J.: The Fiat-Shamir zoo: Relating the security of different signature variants. In: Gruschka, N. (ed.) *Secure IT Systems - 23rd Nordic Conference, NordSec 2018, Oslo, Norway, November 28-30, 2018, Proceedings. Lecture Notes in Computer Science*, vol. 11252, pp. 154–170. Springer (2018). [https://doi.org/10.1007/978-3-030-03638-6\\_10](https://doi.org/10.1007/978-3-030-03638-6_10)
3. Basso, A., Dartois, P., De Feo, L., Leroux, A., Maino, L., Pope, G., Robert, D., Wesolowski, B.: SQISign2D-west: The fast, the small, and the safer. *Cryptology ePrint Archive, Report 2024/760* (2024), <https://eprint.iacr.org/2024/760>
4. Basso, A., Maino, L., Pope, G.: FESTA: Fast encryption from supersingular torsion attacks. In: Guo, J., Steinfeld, R. (eds.) *ASIACRYPT 2023, Part VII. LNCS*, vol. 14444, pp. 98–126. Springer, Singapore (Dec 2023). [https://doi.org/10.1007/978-981-99-8739-9\\_4](https://doi.org/10.1007/978-981-99-8739-9_4)
5. Bernstein, D.J., De Feo, L., Leroux, A., Smith, B.: Faster computation of isogenies of large prime degree. *Open Book Series* **4**(1), 39–55 (2020). <https://doi.org/10.2140/obs.2020.4.39>
6. Biasse, J.F., Jao, D., Sankar, A.: A quantum algorithm for computing isogenies between supersingular elliptic curves. In: Meier, W., Mukhopadhyay, D. (eds.) *INDOCRYPT 2014. LNCS*, vol. 8885, pp. 428–442. Springer, Cham (Dec 2014). [https://doi.org/10.1007/978-3-319-13039-2\\_25](https://doi.org/10.1007/978-3-319-13039-2_25)
7. Castryck, W., Chen, M., Invernizzi, R., Lorenzon, G., Vercauteren, F.: Breaking and repairing SQISign2D-East. *Cryptology ePrint Archive, Paper 2024/1453* (2024), <https://eprint.iacr.org/2024/1453>
8. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) *EUROCRYPT 2023, Part V. LNCS*, vol. 14008, pp. 423–447. Springer, Cham (Apr 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_15](https://doi.org/10.1007/978-3-031-30589-4_15)
9. Chavez-Saab, J., Santos, M.C., De Feo, L., Eriksen, J.K., Hess, B., Kohel, D., Leroux, A., Longa, P., Meyer, M., Panny, L., Patranabis, S., Petit, C., Rodríguez Henríquez, F., Schaeffler, S., Wesolowski, B.: SQISign. Tech. rep., National Institute of Standards and Technology (2023), available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures>
10. Costello, C., Hisil, H.: A simple and compact algorithm for SIDH with arbitrary degree isogenies. In: Takagi, T., Peyrin, T. (eds.) *ASIACRYPT 2017, Part II. LNCS*, vol. 10625, pp. 303–329. Springer, Cham (Dec 2017). [https://doi.org/10.1007/978-3-319-70697-9\\_11](https://doi.org/10.1007/978-3-319-70697-9_11)
11. Dartois, P., Leroux, A., Robert, D., Wesolowski, B.: SQISignHD: New dimensions in cryptography. In: Joye, M., Leander, G. (eds.) *EUROCRYPT 2024, Part I. LNCS*, vol. 14651, pp. 3–32. Springer, Cham (May 2024). [https://doi.org/10.1007/978-3-031-58716-0\\_1](https://doi.org/10.1007/978-3-031-58716-0_1)
12. Dartois, P., Maino, L., Pope, G., Robert, D.: An algorithmic approach to  $(2, 2)$ -isogenies in the theta model and applications to isogeny-based cryptography. *Cryptology ePrint Archive, Report 2023/1747* (2023), <https://eprint.iacr.org/2023/1747>
13. De Feo, L.: Mathematics of isogeny based cryptography (2017), <https://arxiv.org/abs/1711.04062>
14. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: Compact post-quantum signatures from quaternions and isogenies. In: Moriai, S., Wang, H. (eds.) *ASIACRYPT 2020, Part I. LNCS*, vol. 12491, pp. 64–93. Springer, Cham (Dec 2020). [https://doi.org/10.1007/978-3-030-64837-4\\_3](https://doi.org/10.1007/978-3-030-64837-4_3)



15. De Feo, L., Leroux, A., Longa, P., Wesolowski, B.: New algorithms for the deuring correspondence - towards practical and secure SQISign signatures. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 659–690. Springer, Cham (Apr 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_23](https://doi.org/10.1007/978-3-031-30589-4_23)
16. Delfs, C., Galbraith, S.D.: Computing isogenies between supersingular elliptic curves over  $\mathbb{F}_p$ . DCC **78**(2), 425–440 (2016). <https://doi.org/10.1007/s10623-014-0010-1>
17. Deuring, M.: Die Typen der Multiplikatorenringe elliptischer Funktionenkörper. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* **14**, 197–272 (1941), <https://doi.org/10.1007/BF02940746>
18. Dirichlet, P.G.L.: Beweis des Satzes, dass jede unbegrenzte arithmetische Progression, deren erstes Glied und Differenz ganze Zahlen ohne gemeinschaftlichen Factor sind, unendlich viele Primzahlen enthält. *Abhandlungen der Königlich-Preussischen Akademie der Wissenschaften zu Berlin* **48**, 45–71 (1837). <https://doi.org/10.1017/CBO9781139237321.012>
19. Duparc, M., Fouotsa, T.B., Vaudenay, S.: SILBE: an updatable public key encryption scheme from lollipop attacks. *Cryptology ePrint Archive*, Report 2024/400 (2024), <https://eprint.iacr.org/2024/400>
20. Erbsen, A., Philipoom, J., Gross, J., Sloan, R., Chlipala, A.: Simple high-level code for cryptographic arithmetic - with proofs, without compromises. In: 2019 IEEE Symposium on Security and Privacy. pp. 1202–1219. IEEE Computer Society Press (May 2019). <https://doi.org/10.1109/SP.2019.00005>
21. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). [https://doi.org/10.1007/3-540-47721-7\\_12](https://doi.org/10.1007/3-540-47721-7_12)
22. Grover, L.K.: A fast quantum mechanical algorithm for database search. In: 28th ACM STOC. pp. 212–219. ACM Press (May 1996). <https://doi.org/10.1145/237814.237866>
23. Hardy, G.H., Wright, E.M.: *An Introduction to the Theory of Numbers*. Oxford, sixth edn. (1975). <https://doi.org/10.1093/oso/9780199219858.001.0001>
24. Jao, D., Azarderakhsh, R., Campagna, M., Costello, C., De Feo, L., Hess, B., Jalali, A., Koziel, B., LaMacchia, B., Longa, P., Naehrig, M., Renes, J., Soukharev, V., Urbanik, D., Pereira, G., Karabina, K., Hutchinson, A.: SIKE. Tech. rep., National Institute of Standards and Technology (2022), available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions>
25. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: Yang, B.Y. (ed.) *Post-Quantum Cryptography - 4th International Workshop, PQCrypto 2011*. pp. 19–34. Springer, Berlin, Heidelberg (Nov / Dec 2011). [https://doi.org/10.1007/978-3-642-25405-5\\_2](https://doi.org/10.1007/978-3-642-25405-5_2)
26. Kani, E.: The number of curves of genus two with elliptic differentials. *Journal für die reine und angewandte Mathematik* **485**, 93–122 (1997). <https://doi.org/10.1515/crll.1997.485.93>
27. Kirschmer, M., Voight, J.: Algorithmic enumeration of ideal classes for quaternion orders. *SIAM Journal on Computing* **39**(5), 1714–1747 (2010). <https://doi.org/10.1137/080734467>
28. Kohel, D., Lauter, K., Petit, C., Tignol, J.P.: On the quaternion-isogeny path problem. *LMS Journal of Computation and Mathematics* **17**(A), 418–432 (2014). <https://doi.org/10.1112/S1461157014000151>



29. Leroux, A.: Quaternion algebras and isogeny-based cryptography. Ph.D. thesis, École Polytechnique, France (2022), <http://www.lix.polytechnique.fr/Labo/Antonin.LEROUX/manuscrit.these.pdf>
30. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 448–471. Springer, Cham (Apr 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_16](https://doi.org/10.1007/978-3-031-30589-4_16)
31. Nakagawa, K., Onuki, H.: QFESTA: Efficient algorithms and parameters for FESTA using quaternion algebras. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part V. LNCS, vol. 14924, pp. 75–106. Springer, Cham (Aug 2024). [https://doi.org/10.1007/978-3-031-68388-6\\_4](https://doi.org/10.1007/978-3-031-68388-6_4)
32. Nakagawa, K., Onuki, H.: SQIsign2D-east: A new signature scheme using 2-dimensional isogenies. Cryptology ePrint Archive, Report 2024/771 (2024), <https://eprint.iacr.org/2024/771>
33. Onuki, H., Nakagawa, K.: Ideal-to-isogeny algorithm using 2-dimensional isogenies and its application to SQIsign. Cryptology ePrint Archive, Report 2024/778 (2024), <https://eprint.iacr.org/2024/778>
34. Page, A., Robert, D.: Introducing clapoti(s): Evaluating the isogeny class group action in polynomial time. Cryptology ePrint Archive, Report 2023/1766 (2023), <https://eprint.iacr.org/2023/1766>
35. Page, A., Wesolowski, B.: The supersingular endomorphism ring and one endomorphism problems are equivalent. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VI. LNCS, vol. 14656, pp. 388–417. Springer, Cham (May 2024). [https://doi.org/10.1007/978-3-031-58751-1\\_14](https://doi.org/10.1007/978-3-031-58751-1_14)
36. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 387–398. Springer, Berlin, Heidelberg (May 1996). [https://doi.org/10.1007/3-540-68339-9\\_33](https://doi.org/10.1007/3-540-68339-9_33)
37. Renes, J.: Computing isogenies between Montgomery curves using the action of  $(0, 0)$ . In: Lange, T., Steinwandt, R. (eds.) Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018. pp. 229–247. Springer, Cham (2018). [https://doi.org/10.1007/978-3-319-79063-3\\_11](https://doi.org/10.1007/978-3-319-79063-3_11)
38. Robert, D.: Evaluating isogenies in polylogarithmic time. Cryptology ePrint Archive, Report 2022/1068 (2022), <https://eprint.iacr.org/2022/1068>
39. Robert, D.: Breaking SIDH in polynomial time. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 472–503. Springer, Cham (Apr 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_17](https://doi.org/10.1007/978-3-031-30589-4_17)
40. Robert, D.: Fast pairings via biextensions and cubical arithmetic. Cryptology ePrint Archive, Report 2024/517 (2024), <https://eprint.iacr.org/2024/517>
41. Santos, M.C.R., Eriksen, J.K., Meyer, M., Reijnders, K.: AprèsSQI: Extra fast verification for SQIsign using extension-field signing. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part I. LNCS, vol. 14651, pp. 63–93. Springer, Cham (May 2024). [https://doi.org/10.1007/978-3-031-58716-0\\_3](https://doi.org/10.1007/978-3-031-58716-0_3)
42. Silverman, J.H.: The arithmetic of elliptic curves, Graduate texts in mathematics, vol. 106. Springer (1986). <https://doi.org/10.1007/978-0-387-09494-6>
43. Zanon, G.H.M., Simplicio, M.A., Pereira, G.C.C.F., Doliskani, J., Barreto, P.S.L.M.: Faster key compression for isogeny-based cryptosystems. IEEE Transactions on Computers **68**(5), 688–701 (2019). <https://doi.org/10.1109/TC.2018.2878829>



# Extending Class Group Action Attacks via Sesquilinear Pairings

Joseph Macula<sup>(✉)</sup>  and Katherine E. Stange 

University of Colorado Boulder, Boulder, USA  
Joseph.Macula@colorado.edu, kstange@math.colorado.edu

**Abstract.** We introduce a new tool for the study of isogeny-based cryptography, namely pairings which are sesquilinear (conjugate linear) with respect to the  $\mathcal{O}$ -module structure of an elliptic curve with CM by an imaginary quadratic field  $\mathcal{O}$ . We use these pairings to study the security of problems based on the class group action on collections of oriented ordinary or supersingular elliptic curves. This extends work of [CHM+23] and [FFP24].

**Keywords:** Isogeny-based cryptography · Pairings · Elliptic curves

## 1 Introduction

The use of isogeny graphs in cryptography dates to [CLG09, Cou06, RS06]. The latter proposals were for public-key cryptography based on an ordinary isogeny graph. In particular, the class group  $\text{Cl}(\mathcal{O})$  of an order  $\mathcal{O}$  in an imaginary quadratic field  $K$  acts on the set of ordinary elliptic curves over  $\overline{\mathbb{F}}_p$  with CM by  $\mathcal{O}$ . For efficiency, CSIDH was proposed [CLM+18], making use of supersingular curves with an action by the class group of the Frobenius field. More recently, this was generalized to OSDIH [CK20], making use of other imaginary quadratic fields in the endomorphism algebra. Recently, SIDH adaptations based on related ideas have been proposed [BF23]. Our paper concerns *oriented* elliptic curves, which refers to attaching the data of an embedding of a particular imaginary quadratic order  $\mathcal{O}$  into the endomorphism ring. All these public-key proposals are examples of class group actions on oriented curves.

The security of these schemes relies on variants of the Diffie-Hellman problem for the class group action. The security of these problems has drawn a great deal of interest, and not all instances of the problem have so far proven to be secure. If the class group is even, the decisional Diffie-Hellman problem is broken by the use of genus theory [CSV22, CHVW22]. These papers make use of the Weil and Tate pairings to compute certain associated characters. More recently, [CHM+23] makes use of generalizations of Weil and Tate pairings to break certain instances of the class group action problem (i.e., determining which class group element takes one given oriented curve to another) when the discriminant has a large smooth square factor and the degree is known. Pairings have also appeared in the

study of oriented elliptic curves in [LJ13], to navigate the isogeny graph. For other interactions between pairings and isogeny-based cryptography, see [KT19, Rei23].

The attacks in [CHM+23] use pairings to reduce a hidden isogeny problem with known degree for the class group action to the SIDH problem recently broken using higher dimensional abelian varieties [CD23, MMP+23, Rob23b]. In short, if the degree of a secret isogeny  $\phi : E \rightarrow E'$  is known, and it is known that  $\phi P \in \mathbb{Z}P'$  for  $P \in E$  and  $P' \in E'$ , then we can make use of a relationship of the form

$$\langle P, P \rangle^{\deg \phi} = \langle \phi P, \phi P \rangle = \langle kP', kP' \rangle = \langle P', P' \rangle^{k^2}$$

by solving a discrete logarithm to obtain the relationship  $k^2 \equiv \deg \phi \pmod{m}$ , and thereby solve for  $k$ . With this, we (essentially) obtain the image  $\phi P$  of  $P$ , which is the type of information provided in the SIDH problem. The classical SIDH problem (for which we now have efficient methods) requires the image of two basis points, and this provides only one. To close the gap, [CHM+23] uses results of [FFP24] which reduce  $\text{SIDH}_1$ , in which the image of only one torsion point is provided, to classical SIDH, provided the order of the point is square. More recent work presented but not yet available [CDM+24] uses pairings to generalize the SIDH attacks so that torsion images of any sufficiently large subgroup suffice.

These attacks require that the degree of the secret isogeny is known. This is the case in constant-time implementations aimed at preventing side-channel attacks such as those in [CVCCD+19]; see [CHM+23] for more details. Furthermore, in [FFP24, Lemma 14], the authors give a heuristic reduction from the group action problem to the same problem with known degree. **In this paper we will assume throughout that the degree of the secret isogeny is known.**

In this paper we introduce a new tool for understanding these results and pushing such attacks further. In [Sta24], certain new generalized pairings  $\widehat{W}$  and  $\widehat{T}$  (generalizing the usual Weil and Tate pairings) are defined, which are  $\mathcal{O}$ -sesquilinear, meaning that

$$\langle \alpha x, \beta y \rangle = \langle x, y \rangle^{\overline{\alpha}\beta}$$

for  $\alpha, \beta \in \mathcal{O}$ . In particular, they take values in an  $\mathcal{O}$ -module formed by extending scalars from the usual domain  $\mathbb{F}_q^*$ .

In particular, we need now assume only that  $\phi P \in \mathcal{O}P'$  and obtain a relationship

$$\langle P, P \rangle^{\deg \phi} = \langle \phi P, \phi P \rangle = \langle \lambda P', \lambda P' \rangle = \langle P', P' \rangle^{N(\lambda)},$$

where  $\lambda \in \mathcal{O}$ . The new pairings are amenable to a Miller-type effective algorithm for their computation, and carry all the useful properties of the Weil and Tate pairings, especially compatibility with  $\mathcal{O}$ -oriented isogenies.

The paper [CHM+23] provides a taxonomy of known generalized pairings, but all of these are only  $\mathbb{Z}$ -bilinear with image in  $\mathbb{F}_q^*$ .

One important difference of these sesquilinear pairings from the generalized pairings previously considered is their non-degeneracy. In [CHM+23], there is a

classification theorem for *cyclic self-pairings compatible with oriented endomorphisms*. These are functions  $f_m : C \rightarrow \mu_m$  where  $C$  is a cyclic subgroup of  $E[m]$  whose image under  $f_m$  spans  $\mu_m$ , with the following properties:  $f(\lambda P) = f(P)^{\lambda^2}$ ,  $\iota(\sigma)(P) \in C$ , and  $f(\iota(\sigma)P) = f(P)^{N(\iota(\sigma))}$  for  $\iota$  an orientation of a given imaginary quadratic order  $\mathcal{O}$ ,  $\sigma \in \mathcal{O}$ , and  $P \in C$ . They essentially show that such pairings can only be non-trivial for  $m$  dividing the discriminant  $\Delta_{\mathcal{O}}$  of  $\mathcal{O}$ .

The requirement that  $m$  divide  $\Delta_{\mathcal{O}}$  limits the applicability of their attacks on the class group action to situations where the discriminant has a good factorization. We demonstrate that by extending to  $\mathcal{O}$ -sesquilinear pairings, whose domain is not  $\mathbb{Z}$ -cyclic but instead  $\mathcal{O}$ -cyclic, we obtain many more non-trivial self-pairings to work with.

The use of these new  $\mathcal{O}$ -sesquilinear pairings offers several clarifying conceptual advantages, and partially answers several of the open problems posed in [CHM+23]. However, they are not a magic bullet: we show (Theorem 7) that the computation of these pairings is essentially equivalent to the computation of the  $\mathcal{O}$ -orientation, provided discrete logarithms are efficient in  $\mu_m$  (for example, if  $m$  is smooth).

### Conceptual Contributions

1. We introduce the new  $\mathcal{O}$ -sesquilinear pairing  $\widehat{T}$  in the cryptographic context.
2. We show that these pairings give rise to many non-degenerate  $\mathcal{O}$ -cyclic self-pairings, without a requirement that  $m$  divide the discriminant (Theorem 6).
3. We characterize elliptic curves for which  $E[m]$  is a cyclic  $\mathcal{O}$ -module (Theorem 3):  $E[m]$  is  $\mathcal{O}$ -cyclic if and only if the  $\mathcal{O}$ -orientation is  $m$ -primitive.
4. We show an equivalence between computation of an  $\mathcal{O}$ -orientation and the computation of  $\mathcal{O}$ -sesquilinear pairings for nice  $m$  (Theorem 7).
5. Corollary 1 and Theorem 9 (described in more detail below) provide evidence for a trade-off between the amount of known level structure of a secret isogeny  $\phi : E \rightarrow E'$  of degree  $d$  and how much of the endomorphism rings of  $E$  and  $E'$  we need to represent to find  $\phi$ . As shown in [Wes22], the fixed-degree isogeny problem with full level structure is equivalent to finding a representation of the full endomorphism ring of  $E$  and  $E'$ , while [CD23, MMP+23, Rob23b] show that the fixed-degree isogeny problem with minimal level structure requires no knowledge of even a partial representation of the endomorphism rings of  $E$  and  $E'$ . As described in *Cryptographic contributions* items 2 and 3 below, knowledge of an intermediate level structure can be combined with a representation of only “half” of the endomorphism rings of  $E$  and  $E'$ , to provide attacks on hidden isogenies of known degree. See also the work in [FFP24], which explores varying amounts of level structure.

### Cryptographic Contributions

1. We extend the applicability of the (sometimes polynomial) attacks from [CHM+23] on the class group action problem (Sect. 8). These attacks run for smooth  $m$  dividing the discriminant. We recover these attacks using the

new pairings in a slightly different way, with the advantage that our pairing computations do not require going to a large field extension. This partially addresses one of the open questions of [CHM+23, Section 7]. Example 3 gives an explicit situation in which the reach of polynomial attacks is strictly extended.

2. We demonstrate a pairing-based reduction from  $\text{SIDH}_1$  to  $\text{SIDH}$  in the oriented situation for  $E[m]$ , where  $m$  is smooth and coprime to the discriminant (Theorem 9), resulting in an attack when  $m^2 > \deg \phi$ . This partially addresses the first and second open problems in [CHM+23, Section 7]. Existing attacks on  $\text{SIDH}_1$  (which apply without orientation information) require  $m > \deg \phi$ .
3. We reduce the hard problem underlying FESTA [BMP23] to finding an orientation of the secret isogeny  $\phi : E \rightarrow E'$  (i.e. an orientation of both curves and the isogeny between them) (Corollary 1). This follows from an attack on the Diagonal  $\text{SIDH}$  Problem (Theorem 10).
4. We show how these pairings, using orientation information, easily reveal *partial* information on the image of a torsion point  $P$  of order  $m$  for  $m$  smooth (Theorem 8). This results in an algorithm to break class-group-based schemes by running the  $\text{SIDH}$  attack on  $\sqrt{\deg(\phi)}$  candidate torsion points as images under  $\phi$  (Remark 5).
5. Our results should be considered a cautionary tale for the design of decisional problems based on torsion point images, such as in [MOT20], since the possible images of torsion points is restricted. We discuss this in Remark 6.
6. In the supersingular case, we demonstrate a method of finding the secret isogeny in the presence of two independent known orientations (which amounts to an explicit subring of the endomorphism ring of rank 4), provided the secret isogeny is oriented for both orientations. This is not a surprise, as this problem could be solved by the KLPT algorithm if the endomorphisms are obtained by walking the graph (see [EHL+20], and also [KLPT14, Wes22]), but it provides a new method via a simple reduction to the  $\text{SIDH}$  problem. (Section 9.)

## 2 Background

### 2.1 Notations

We study elliptic curves, typically denoted  $E$ ,  $E'$  etc., defined over finite fields, denoted by  $\mathbb{F}$  in general. Denote an algebraic closure of  $\mathbb{F}$  by  $\overline{\mathbb{F}}$ . The identity on  $E$  is denoted  $\infty$ , and  $\text{End}(E)$  is the endomorphism ring over  $\overline{\mathbb{F}}$ . We study imaginary quadratic fields, denoted  $K$  in general, and orders in such fields, denoted by  $\mathcal{O}$ ,  $\mathcal{O}'$  etc. Greek letters typically denote elements of the orders. We denote the norm of an element  $\lambda$  of a given order by  $N(\lambda)$ . When considering the action of an element  $\alpha \in \mathcal{O}$  on a point  $P$ , we write  $[\alpha]P$ . The Greek letter  $\phi$  always refers to an isogeny and  $\hat{\phi}$  always denotes the dual isogeny of  $\phi$ . For ease of notation, we write  $\phi P$  instead of  $\phi(P)$ . Throughout the paper, we write  $\mu_m$  for the copy of  $\mu_m$  in a finite field.

## 2.2 Orientations

We study  $\mathcal{O}$ -oriented elliptic curves over finite fields, which are curves together with the information of an embedding  $\iota : \mathcal{O} \rightarrow \text{End}(E)$ . This extends to an embedding of the same name,  $\iota : K \rightarrow \mathbb{Q} \otimes_{\mathbb{Z}} \text{End}(E)$ , and the  $\mathcal{O}$ -orientation is called *primitive* if  $\iota(K) \cap \text{End}(E) = \iota(\mathcal{O})$ . If the index  $[\iota(K) \cap \text{End}(E) : \iota(\mathcal{O})]$  is coprime to  $n$ , we say the orientation is *n-primitive*. Given an  $\mathcal{O}$ -orientation, there is a unique  $\mathcal{O}' \supseteq \mathcal{O}$  for which  $\iota$  becomes a  $\mathcal{O}'$ -primitive orientation, namely  $\iota(\mathcal{O}') = \iota(K) \cap \text{End}(E)$ . Given an elliptic curve  $E$  with an  $\mathcal{O}$ -orientation, we define the *relative conductor* of  $\mathcal{O}$  to be the index  $[\mathcal{O}' : \mathcal{O}]$ , for which the orientation is  $\mathcal{O}'$ -primitive.

If  $\phi : E \rightarrow E'$  is an isogeny between two  $\mathcal{O}$ -oriented elliptic curves  $(E, \iota)$  and  $(E', \iota')$  is such that  $\phi \circ \iota(\alpha) = \iota'(\alpha) \circ \phi$  for all  $\alpha \in \mathcal{O}$ , then we say that  $\phi$  is an oriented isogeny. Throughout the paper, we will generally fix a single  $\mathcal{O}$ -orientation for any curve, so we will often drop the  $\iota$ , writing simply  $[\alpha]$  for  $\iota(\alpha)$ , writing  $\mathcal{O} \subseteq \text{End}(E)$ , and characterizing oriented isogenies as those for which  $\phi \circ [\alpha] = [\alpha] \circ \phi$ . This saves on notation.

## 2.3 Cyclic Self-pairings

CSIDH, introduced in [CLM+18], relies for its security on the presumed hardness of the following instance of the *vectorization problem*: given a (large) prime  $p \equiv 3 \pmod{4}$  and two supersingular curves  $E$  and  $E'$  over  $\mathbb{F}_p$ , find the ideal class  $[\mathfrak{a}]$  of  $\mathcal{O} = \mathbb{Z}[\sqrt{-p}]$  such that  $E' = [\mathfrak{a}]E$ . More generally, the vectorization problem can be phrased as follows: given two supersingular elliptic curves  $E, E'$  primitively oriented by an imaginary quadratic order  $\mathcal{O}$  and known to be connected by the action of the ideal class group  $\text{cl}(\mathcal{O})$  of  $\mathcal{O}$ , find  $[\mathfrak{a}] \in \text{cl}(\mathcal{O})$  such that  $E' = [\mathfrak{a}]E$ . This is also known as a class-group-action problem.

This paper builds on the previous work of [CHM+23]. There, the authors ask whether the attack on SIDH [CD23, MMP+23, Rob23b] that renders the protocol insecure can be applied to solving the vectorization problem. In brief, they note that one can treat the SIDH attack as an oracle, which when given the degree  $d$  of a secret  $\mathbb{F}_q$ -rational isogeny  $\phi$  between curves  $E$  and  $E'$  defined over  $\mathbb{F}_q$  and knowledge of its action on a basis of  $E[m]$  for  $m$  coprime to  $d$  and  $m^2 > 4d$ , returns  $\phi$ . Assuming the degree  $d$  of  $\phi$  is known, the question of whether this oracle can answer the vectorization problem therefore reduces to the question of whether one can determine the action of  $\phi$  on a basis of  $E[m]$  for suitable  $m$ .

The following example, reproduced directly from [CHM+23], is instructive. Assume the context of CSIDH, i.e., that the relevant order is  $\mathbb{Z}[\sqrt{-p}]$ . Choosing  $m$  to be the power of a small prime  $l$  coprime to  $d$  that splits in  $\mathbb{Q}(\sqrt{-p})$ ,  $E[m]$  has a basis  $\{P, Q\}$  consisting of eigenvectors of the Frobenius endomorphism  $\pi_p$ . Since by assumption the curves  $E, E'$  and the isogeny  $\phi$  are all defined over  $\mathbb{F}_p$ ,  $E'[m]$  has a basis  $\{P', Q'\}$  consisting of eigenvectors of  $\pi_p$  and  $\phi P = rP'$ ,

$\phi Q = sQ'$  for  $r, s \in (\mathbb{Z}/m\mathbb{Z})^\times$ . The bilinearity and compatibility with isogenies of the  $m$ -Weil pairing imply that

$$e_m(P', Q') = e_m(P, Q)^{rsd} \tag{1}$$

With Miller’s algorithm [Mil04], computation of the  $m$ -Weil pairing is efficient. Since  $m$  is a power of a small prime, also efficient is computation of discrete logarithms. Thus, given knowledge of  $d$  it remains to determine one of  $r$  or  $s$ . Yet properties of the  $m$ -Weil pairing imply that  $e_m(P', P') = 1$ ; thus, one cannot compute  $e_m(P', P') = e_m(P, P)^{r^2d}$  to find  $r$ .

In [CHM+23], this obstacle is surmounted via the construction of cyclic self-pairings that are compatible with  $\mathcal{O}$ -oriented isogenies. A cyclic self-pairing is a function  $f$  defined on a finite cyclic subgroup  $C$  of an elliptic curve  $E/\mathbb{F}$  with the property that

$$f(rP) = f(P)^{r^2} \text{ for all } P \in C \text{ and } r \in \mathbb{Z}.$$

When  $E$  and  $E'$  are two curves over  $\mathbb{F}$  with orientations of an imaginary quadratic order  $\mathcal{O}$  by  $\iota, \iota'$ , respectively, two self-pairings  $f$  and  $f'$  defined on finite subgroups  $C$  of  $E$  and  $C'$  of  $E'$  are compatible with  $\mathcal{O}$ -oriented isogenies  $\phi : E \rightarrow E'$  when  $\phi(C) \subset C'$  and  $f'(\phi P) = f(P)^{\deg \phi}$ . If  $\phi$  is an  $\mathcal{O}$ -oriented isogeny from  $E$  to  $E'$  of degree  $d$  coprime to an integer  $m$  such that  $E$  and  $E'$  have non-trivial cyclic self-pairings  $f$  and  $f'$  compatible with  $\mathcal{O}$ -oriented isogenies on cyclic subgroups  $\langle P \rangle, \langle P' \rangle$ , then

$$f'(P') = f(P)^{dr^2}$$

for some  $r \in \mathbb{Z}/m\mathbb{Z}$ . If furthermore discrete logarithms are efficiently computable modulo  $m$ , then the non-triviality of  $f$  and  $f'$  implies that one can efficiently determine  $r^2$  modulo  $m$ . Assuming  $m$  has a nice factorization, this leaves only a few possibilities for  $r$ , and one simply guesses by direct computation which one is correct.

The non-trivial self-pairings constructed in [CHM+23] are generalizations on the classical reduced  $m$ -Tate pairing. We refer the reader to Sect. 5 of [CHM+23] for further details. Crucially, the order  $m$  of non-trivial cyclic self-pairings compatible with  $\mathcal{O}$ -oriented isogenies must divide  $\Delta_{\mathcal{O}}$  ([CHM+23], Proposition 4.8). Furthermore, the existence of such a self-pairing only yields knowledge of the image of a single torsion point  $P$  under  $\phi$ . In [CHM+23], this latter issue is addressed by assuming  $m^2 \mid \Delta_{\mathcal{O}}$ . Then one obtains the image of an order  $m^2$  point  $P$  under  $\phi$ . The authors briefly describe how one can then obtain from this data knowledge of the action of  $\phi$  on a basis for  $E[m]$ . A more systematic description of this reduction in the language of level structure is in Sect. 5 of [FFP24] (in particular, see corollary 12).

Thus, there are two significant limitations to the scope of this attack. First,  $\Delta_{\mathcal{O}}$  must contain a large smooth square factor. Second, in general one must work over a field where the  $m^2$ -torsion is fully rational. In the worst case, this requires a base change to an extension of potentially large degree.

The first of these limitations is addressed in work in preparation by Castryck et al. [CDM+24], the authors show that with knowledge of the image of  $\phi : E \rightarrow E'$  on a generating set  $S$  for a subgroup  $G$  with  $\#G > 4d$ , there is an algorithm to determine  $\phi$  (in the sense of computing arbitrary images) that is polynomial in

- (1) the size of the  $S$  and  $\log q$ , where  $q$  is the size of the field over which  $E$  and  $E'$  are defined;
- (2) the size of the largest prime factor of  $\#G$ ;
- (3) the largest degree of the fields of definition of  $E[\ell^{\lfloor e/2 \rfloor}]$ , taken over all prime powers  $l^e$  dividing  $\#G$ .

In particular, this result obviates the need for  $\Delta_{\mathcal{O}}$  to contain a smooth square factor; instead, one only needs a smooth factor of size greater than  $4d$ .

### 2.4 Level Structure

Many isogeny-based protocols require that some torsion-point information be made publicly known. For example, in SIDH, the image under the secret isogeny  $\phi$  of a specified basis  $\{P, Q\}$  for  $E[m]$  (where  $m$  is a power of a small prime coprime to the characteristic of the field  $\mathbb{F}$  over which  $E$  is defined) is known. As discussed in the last section, in CSIDH the image of a basis  $\{P, Q\}$  for  $E[m]$  (with  $m$  again a power of a small prime coprime to the field characteristic, but also coprime to the degree  $d$  of the secret isogeny  $\phi$ ) is known, up to multiplication by an element of  $(\mathbb{Z}/m\mathbb{Z})^\times$ . Equivalently, the image under  $\phi$  of two order  $m$  subgroups of  $E$  is known. Both types of torsion-point information are examples of *level structure* that  $\phi$  respects.

**Definition 1 ([FFP24]).** *Let  $E$  be an elliptic curve over a finite field  $\mathbb{F}$  of characteristic  $p$  and  $m$  be a positive integer coprime to  $p$ . Let  $\Gamma$  be a subgroup of  $\text{GL}_2(\mathbb{Z}/m\mathbb{Z})$ . A  $\Gamma$ -level structure of level  $m$  on  $E$  is a  $\Gamma$ -orbit of a basis of  $E[m]$ .*

Typically in the context of isogeny problems, one is not interested in level structure *per se*, but in level structure that a given isogeny  $\phi$  respects. That is, given curves  $E$  and  $E'$  both with  $\Gamma$ -level  $m$  structures for a fixed  $\Gamma$ ,  $\phi$  maps the specified  $\Gamma$ -orbit for  $E[m]$  to the specified  $\Gamma$ -orbit for  $E'[m]$ .

There has been much attention paid recently to elliptic curves equipped with a particular level structure. Arpin [Arp24] studies the correspondence of Eichler orders in the quaternion algebra  $B_{p,\infty}$  with supersingular elliptic curves over  $\overline{\mathbb{F}}_p$  equipped with *Borel* level structure—i.e., where  $\Gamma = (\{ \begin{smallmatrix} * & 0 \\ * & * \end{smallmatrix} \})$ —for  $m$  squarefree and coprime to  $p$ . In [CL24], the authors consider the structure of the supersingular isogeny graph with varying level structures and show that many of these graphs remain Ramanujan. Others investigate the actions of generalized ideal class groups on elliptic curves over finite fields with level structure [GPV23, ACKE+24]. In this paper, we are primarily interested in level structure



as a unifying framework for understanding the security of various proposals in isogeny-based cryptography. This framework is described in detail in [FFP24]. In particular, those authors make explicit the implicit level structures in several schemes including SIDH, M-SIDH, CSIDH, and FESTA, and prove several security reductions between various level structures.

### 2.5 Computational Assumptions

With regards to computations, we use the word *efficient* to mean polynomial time in the size of the input, which is itself typically a polynomial in  $\log m$  (the torsion) and  $\log q$  (where  $q$  is the cardinality of the field of definition of the  $m$ -torsion), in our context. Throughout the paper, when we assume that we are given an  $\mathcal{O}$ -oriented elliptic curve, we mean that we are given an explicit orientation, and in particular that, given an element  $\alpha \in \mathcal{O}$ , we can compute its action  $[\alpha]$  on a point  $P$  on  $E$  efficiently.

We assume throughout that the degree of the hidden isogeny is known.

We assume that  $m$  is coprime to the characteristic  $p$  of the given field  $\mathbb{F}$ , and that  $m$  is smooth, meaning that its factors are polynomial in size, so that discrete logarithms in  $\mu_m$  or  $E[m]$  are computable in polynomial time. In particular, we can efficiently write any element of  $E[m]$  in terms of a given basis.

### 2.6 The Tate-Lichtenbaum Frey-Rück Pairing

We review the definition and basic properties of the Tate-Lichtenbaum pairing.

**Definition 2.** *Let  $m > 1$  be an integer. Let  $E$  be an elliptic curve defined over a field  $\mathbb{F}$  (assumed finite in this paper). Suppose that  $P \in E(\mathbb{F})[m]$ . Choose divisors  $D_P$  and  $D_Q$  of disjoint support such that  $D_P \sim (P) - (\mathcal{O})$  and  $D_Q \sim (Q) - (\mathcal{O})$ . Then  $mD_P \sim \emptyset$ , hence there is a function  $f_P$  such that  $\text{div}(f_P) = mD_P$ . The Tate-Lichtenbaum pairing*

$$t_m : E(\mathbb{F})[m] \times E(\mathbb{F})/mE(\mathbb{F}) \rightarrow \mathbb{F}^*/(\mathbb{F}^*)^m$$

is defined by

$$t_m(P, Q) = f_P(D_Q).$$

The standard properties of the Tate pairing are as follows. Proofs can be found in many places, for example [Rob23a] and [CHM+23, Sec 3.2].

**Proposition 1.** *Definition 2 is well-defined, and has the following properties:*

1. *Bilinearity: for  $P, P' \in E(\mathbb{F})[m]$  and  $Q, Q' \in E(\mathbb{F})$*

$$\begin{aligned} t_m(P + P', Q) &= t_m(P, Q)t_m(P', Q), \\ t_m(P, Q + Q') &= t_m(P, Q)t_m(P, Q'). \end{aligned}$$

2. *Non-degeneracy:* Let  $\mathbb{F}$  be a finite field containing the  $m$ -th roots of unity  $\mu_m$ . For nonzero  $P \in E(\mathbb{F})[m]$ , there exists  $Q \in E(\mathbb{F})$  such that

$$t_m(P, Q) \neq 1.$$

Furthermore, for  $Q \in E(\mathbb{F}) \setminus mE(\mathbb{F})$ , there exists a  $P \in E(\mathbb{F})[m]$  such that

$$t_m(P, Q) \neq 1.$$

In particular, for  $P$  of order  $m$ , there exists  $Q$  such that  $t_m(P, Q)$  has order  $m$ , and similarly for the other entry.

3. *Compatibility:* For a point  $P \in E(\mathbb{F})[m]$ , an isogeny  $\phi : E \rightarrow E'$ , and a point  $Q \in E'(\mathbb{F})$ ,

$$t_m(\phi P, Q) = t_m(P, \widehat{\phi}Q).$$

### 3 Structure of $E[\alpha]$

Suppose  $E$  has an  $\mathcal{O}$ -orientation. Let  $\alpha \in \mathcal{O}$ . We wish to know when  $E[\alpha]$  is cyclic as an  $\mathcal{O}$ -module. The following two theorems of Lenstra are relevant.

**Theorem 1 ([Len96, Proposition 2.1]).** *Let  $E$  be an elliptic curve over an algebraically closed field  $k$ , and  $\mathcal{O}$  a subring of  $\text{End}_k(E)$  such that as  $\mathbb{Z}$ -modules,  $\mathcal{O}$  is free of rank 2 and  $\text{End}_k(E)/\mathcal{O}$  is torsion-free. Then for every separable element  $\alpha \in \mathcal{O}$ ,  $E[\alpha] \cong \mathcal{O}/\alpha\mathcal{O}$  as  $\mathcal{O}$ -modules.*

When  $\alpha$  is inseparable, Lenstra has a similar result. With  $\mathcal{O}$  as above,  $\text{char } k = p > 0$ , and  $K = \mathcal{O} \otimes_{\mathbb{Z}} \mathbb{Q}$ , he observes that there is a  $p$ -adic valuation  $\nu$  on  $K$  with  $\nu(\alpha) = \log(\deg_i \alpha) / \log p$  for  $\alpha \in \mathcal{O}$ . Following his notation, we define  $V = \{x \in K : \nu(x) \geq 0\}$ .

**Theorem 2 ([Len96, Proposition 2.4]).** *Let the notation and hypotheses be as above. Then for every non-zero element  $\alpha \in \mathcal{O}$  there is an isomorphism of  $\mathcal{O}$ -modules  $E[\alpha] \oplus (V/\alpha V) \cong \mathcal{O}/\alpha\mathcal{O}$ .*

**Theorem 3.** *Let  $E$  be an elliptic curve over  $\mathbb{F}$ ,  $K$  an imaginary quadratic field, and  $\mathcal{O}$  an order in  $K$  such that  $E$  has an  $\mathcal{O}$ -orientation, which is primitive when extended to  $\mathcal{O}'$ . Let  $f = [\mathcal{O}' : \mathcal{O}]$  be the relative conductor of  $\mathcal{O}$ . For  $\alpha \in \mathcal{O}$  with  $N(\alpha)$  coprime to  $f$  (i.e., such that the  $\mathcal{O}$ -orientation is  $N(\alpha)$ -primitive), then  $E[\alpha]$  is cyclic as an  $\mathcal{O}$ -module. Specifically:*

1. *If  $\alpha$  is separable, then  $E[\alpha] \cong \mathcal{O}/\alpha\mathcal{O}$ .*
2. *If  $\alpha$  is inseparable, then  $E[\alpha]$  is isomorphic to a proper cyclic  $\mathcal{O}$ -submodule of  $\mathcal{O}/\alpha\mathcal{O}$ .*

As a partial converse, as soon as  $\alpha$  factors through multiplication by  $n$  for some  $n > 1$  that divides  $f$ ,  $E[\alpha]$  is not cyclic as an  $\mathcal{O}$ -module. In particular, if  $\alpha = m \in \mathbb{Z}$ , then  $E[m]$  is a cyclic  $\mathcal{O}$ -module if and only if  $m$  and  $f$  are coprime.

*Proof.* Suppose first that  $\alpha$  is separable. Let  $\iota$  be an  $\mathcal{O}$ -orientation for  $E$  and  $\mathcal{O}'$  be the order of  $K$  for which  $\iota$  is a primitive orientation. Then as an abelian group the quotient  $\text{End}(E)/\mathcal{O}'$  is torsion-free and Theorem 1 tells us that  $E[\alpha] \cong \mathcal{O}'/\alpha\mathcal{O}'$  as  $\mathcal{O}'$ -modules. We have  $N(\alpha) = N(\alpha\mathcal{O}') = |\mathcal{O}'/\alpha\mathcal{O}'|$ , so since  $N(\alpha)$  is coprime to  $f$ , it follows from [Cox22, Proposition 7.18, 7.20] that the natural injection  $\mathcal{O}/\alpha\mathcal{O} \rightarrow \mathcal{O}'/\alpha\mathcal{O}'$  is an isomorphism of  $\mathcal{O}$ -modules.

Suppose then that  $\alpha$  is inseparable. Let  $\mathcal{O}'$  be as above. From Theorem 2 we have  $E[\alpha] \oplus V/\alpha V \cong \mathcal{O}'/\alpha\mathcal{O}'$ , so  $E[\alpha]$  is isomorphic as an  $\mathcal{O}'$ -module to  $(\mathcal{O}'/\alpha\mathcal{O}')/(V/\alpha V)$  and hence is a cyclic  $\mathcal{O}'$ -module. Since  $\mathcal{O}'/\alpha\mathcal{O}' \cong \mathcal{O}/\alpha\mathcal{O}$  as  $\mathcal{O}$ -modules, again by our assumption that  $N(\alpha)$  is coprime to  $f$ , it follows that  $E[\alpha]$  is cyclic as an  $\mathcal{O}$ -module.

Finally, suppose  $\alpha$  factors through  $[n]$  for some  $n > 1$  with  $n \mid f$ . Then as a  $\mathbb{Z}$ -module,  $E[\alpha] \cong \mathbb{Z}/b\mathbb{Z} \times \mathbb{Z}/c\mathbb{Z}$  with  $n \mid b \mid c$ . Let  $Q$  be an arbitrary point of order  $c$  in  $E[\alpha]$  and extend to a generating set  $\{P, Q\}$  for  $E[\alpha]$  with  $\text{ord}(P) = b, \text{ord}(Q) = c$ . Let  $\mathcal{O}' = \mathbb{Z}[\sigma]$  for some  $\sigma$ . Then  $\mathcal{O} = \mathbb{Z}[f\sigma]$ . Since any element of  $\mathcal{O}$  is a  $\mathbb{Z}$ -linear combination of  $[1]$  and  $[f\sigma]$ , whether or not  $E[\alpha]$  is cyclic as an  $\mathcal{O}$ -module is determined by the action of  $f\sigma$ . We have  $[f\sigma]Q = [n\sigma]Q'$ , where  $Q' = [f/n]Q$ .

If  $[\sigma]Q' = [s]P + [t]Q$ , then  $[f\sigma]Q = [ns]P + [nt]Q$ , and we cannot obtain  $P$  from the action of any  $\mathbb{Z}$ -linear combination of  $[1]$  and  $[f\sigma]$  on  $Q$  (since  $[n]$  is not injective on  $E[\alpha]$ ). Thus,  $Q$  cannot be a generator for  $E[\alpha]$  as an  $\mathcal{O}$ -module. Since  $Q$  was an arbitrary order  $c$  point, and since no point of order strictly less than  $c$  can generate  $E[\alpha]$  as an  $\mathcal{O}$ -module (endomorphisms send points of  $E$  of order  $m$  to points of  $E$  of order dividing  $m$ ),  $E[\alpha]$  cannot be a cyclic  $\mathcal{O}$ -module. □

*Example 1.* Consider the ordinary curve  $y^2 = x^3 + 30x + 2$  over  $\mathbb{F}_{101}$ . Denoting the Frobenius endomorphism of degree  $p$  by  $\pi$ ,  $\mathbb{Z}[\pi]$  has conductor 2 in the maximal order and  $[\mathbb{Z}[\pi] : \mathbb{Z}[\pi^2]] = 18$ . Thus, Theorem 3 implies  $E[3]$  is not cyclic as a  $\mathbb{Z}[\pi^2]$ -module. Indeed, making a base change to  $\mathbb{F}_{101^2}$ , the 3-torsion of  $E$  becomes rational. On the other hand,  $E[3]$  is cyclic as a  $\mathbb{Z}[\pi]$ -module. With  $\mathbb{F}_{101^2} = \mathbb{F}_{101}(a)$  and  $x^2 - 4x + 2$  the minimal polynomial of  $a$ , we have  $P = (41a + 16, 39a + 19) \in E[3]$  and  $\pi(P) = (60a + 79, 62a + 74) \notin \langle P \rangle$ , hence  $\mathbb{Z}[\pi]P = E[3]$ .

### 4 Sesquilinear Pairings

We follow [Sta24] in this section. Suppose  $\mathcal{O} = \mathbb{Z}[\tau]$  is an imaginary quadratic order. Let  $E$  have CM by  $\mathcal{O}$ . Let  $\rho : \mathcal{O} \rightarrow M_{2 \times 2}(\mathbb{Z})$  be the left-regular representation of  $\mathcal{O}$  acting on the basis 1 and  $\tau$ , i.e.

$$\rho(\alpha) = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \iff \alpha = a + c\tau, \quad \alpha\tau = b + d\tau.$$

Then we endow the Cartesian square  $(\mathbb{F}^*)^{\times 2}$  of the multiplicative  $\mathbb{Z}$ -module  $\mathbb{F}^*$  (i.e.  $\mathbb{Z}$ -coefficients in the exponent) with a multiplicative  $\mathcal{O}$ -module action (i.e.

$\mathcal{O}$ -coefficients in the exponent) via

$$(x, y)^\alpha = \rho(\alpha) \cdot (x, y) = (x^a y^b, x^c y^d), \quad \text{where } \rho(\alpha) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}. \quad (2)$$

In the case of an  $\mathcal{O}$ -module, by *order* of an element we mean the  $\mathbb{Z}$ -order; we can also discuss the annihilator as an  $\mathcal{O}$ -module, which may be distinct from this.

For each  $\alpha \in \mathcal{O}$ , we define a bilinear pairing

$$\widehat{T}_\alpha^\tau : E[\overline{\alpha}](\mathbb{F}) \times E(\mathbb{F})/[\alpha]E(\mathbb{F}) \rightarrow (\mathbb{F}^*)^{\times 2}/((\mathbb{F}^*)^{\times 2})^\alpha$$

as follows. Write

$$\rho(\alpha) = \begin{pmatrix} a & b \\ c & d \end{pmatrix}, \quad \rho(\overline{\alpha}) = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}.$$

Observe that this corresponds to the ring facts

$$a + c\tau = \alpha, \quad b + d\tau = \alpha\tau, \quad d - c\tau = \overline{\alpha}, \quad -b + a\tau = \overline{\alpha}\tau.$$

We take  $P \in E[\overline{\alpha}]$ , Define  $f_P = (f_{P,1}, f_{P,2})$ , where

$$\begin{aligned} \operatorname{div}(f_{P,1}) &= a[-\tau]P + b(P) - (a + b)(\infty), \\ \operatorname{div}(f_{P,2}) &= c[-\tau]P + d(P) - (c + d)(\infty). \end{aligned}$$

Choose an auxiliary point  $R \in E(\overline{\mathbb{F}})$  and define for  $Q \in E(\mathbb{F})$ ,

$$D_{Q,1} = ([-\tau]Q + [-\tau]R) - ([-\tau]R), \quad D_{Q,2} = (Q + R) - (R).$$

Then, choosing  $R$  so that the necessary supports are disjoint (i.e. the support of  $\operatorname{div}(f_{P,i})$  and  $D_{Q,j}$  are disjoint for each pair  $i, j$ ), the pairing is defined (using (2)) as

$$\widehat{T}_\alpha^\tau(P, Q) := (f_{P,1}(D_{Q,1}), f_{P,2}(D_{Q,1})) (f_{P,1}(D_{Q,2}), f_{P,2}(D_{Q,2}))^\tau$$

which can also be expressed as

$$(f_{P,1}(D_{Q,1})f_{P,1}(D_{Q,2})^{Tr(\tau)}f_{P,2}(D_{Q,2})^{N(\tau)}, f_{P,2}(D_{Q,1})f_{P,1}(D_{Q,2})^{-1}).$$

*Remark 1.* In [Sta24], it is shown how it is possible to think of these definitions as elements of  $\mathcal{O} \otimes_{\mathbb{Z}} \operatorname{Pic}^0(E)$ :

$$D_Q = D_{Q,1} + \tau \cdot D_{Q,2}, \quad D_P = ([-\tau]P) - (\infty) + \tau \cdot ((P) - (\infty)),$$

and analogously define  $f_P$  satisfying  $\operatorname{div}(f_P) = \alpha \cdot D_P$ , so that the definition above has the form  $f_P(D_Q)$  as for the classical Tate pairing, and the apparent dependence on the basis  $1, \tau$  disappears. For simplicity here, we stick to the direct definition above. In that same paper, analogous constructions are also given for quaternion orders and Weil-like pairings.

**Theorem 4 ([Sta24, Theorems 5.4, 5.5, 5.6]).** *The pairing above is well-defined and satisfies*

1. *Sesquilinearity:* For  $P \in E[\bar{\alpha}](\mathbb{F})$  and  $Q \in E(\mathbb{F})$ ,

$$\widehat{T}_\alpha^\tau([\gamma]P, [\delta]Q) = \widehat{T}_\alpha^\tau(P, Q)^{\bar{\gamma}\bar{\delta}}.$$

2. *Compatibility:* Let  $\phi : E \rightarrow E'$  be an isogeny between curves with CM by  $\mathcal{O}$  and satisfying  $[\alpha] \circ \phi = \phi \circ [\alpha]$ . Then for  $P \in E[\bar{\alpha}](\mathbb{F})$  and  $Q \in E(\mathbb{F})$ ,

$$\widehat{T}_\alpha^\tau(\phi P, \phi Q) = \widehat{T}_\alpha^\tau(P, Q)^{\deg \phi}.$$

3. *Coherence:* Suppose  $P \in E[\bar{\alpha}\bar{\beta}](\mathbb{F})$ , and  $Q \in E(\mathbb{F})/[\alpha\beta]E(\mathbb{F})$ . Then

$$\widehat{T}_{\alpha\beta}^\tau(P, Q) \bmod ((\mathbb{F}^*)^{\times 2})^\beta = \widehat{T}_\beta^\tau([\bar{\alpha}]P, Q \bmod [\beta]E).$$

Suppose  $P \in E[\bar{\alpha}](\mathbb{F})$ , and  $Q \in E(\mathbb{F})/[\alpha\beta]E(\mathbb{F})$ . Then

$$\widehat{T}_{\alpha\beta}^\tau(P, Q) \bmod ((\mathbb{F}^*)^{\times 2})^\alpha = \widehat{T}_\alpha^\tau(P, [\beta]Q \bmod [\alpha]E).$$

4. *Non-degeneracy:* Let  $\mathbb{F}$  be a finite field, and let  $E$  be an elliptic curve defined over  $\mathbb{F}$ . Let  $\alpha \in \mathcal{O}$  be coprime to  $\text{char}(\mathbb{F})$  and the discriminant of  $\mathcal{O}$ . Let  $N = N(\alpha)$ . Suppose  $\mathbb{F}$  contains the  $N$ -th roots of unity. Suppose there exists  $P \in E[N](\mathbb{F})$  such that  $\mathcal{O}P = E[N] = E[N](\mathbb{F})$ . Then

$$\widehat{T}_\alpha^\tau : E[\bar{\alpha}](\mathbb{F}) \times E(\mathbb{F})/[\alpha]E(\mathbb{F}) \rightarrow (\mathbb{F}^*)^{\times 2}/((\mathbb{F}^*)^{\times 2})^\alpha,$$

is non-degenerate. Furthermore, if  $P$  has annihilator  $\bar{\alpha}\mathcal{O}$ , then  $T_\alpha(P, \cdot)$  is surjective; and if  $Q$  has annihilator  $\alpha\mathcal{O}$ , then  $T_\alpha(\cdot, Q)$  is surjective.

5. Let  $t_n$  be the  $n$ -Tate-Lichtenbaum pairing as described in Sect. 2. Then

$$\widehat{T}_n^\tau(P, Q) = \left( t_n(P, Q)^{2N(\tau)} t_n([- \tau]P, Q)^{Tr(\tau)}, t_n([\tau - \bar{\tau}]P, Q) \right).$$

6. Provided both of the following quantities are defined,

$$\widehat{T}_{N(\alpha)}^\tau(P, Q) = \widehat{T}_\alpha^\tau(P, Q)^{\bar{\alpha}} \pmod{((\mathbb{F}^*)^{\times 2})^\alpha}$$

**Theorem 5.** *Provided the action of  $\tau$  is efficiently computable, then the pairing  $\widehat{T}_\alpha^\tau(P, Q)$  is efficiently computable. That is, it takes polynomially many operations in the field of definition of  $P$  and  $Q$ .*

*Proof.* This follows from the definition given above, which is amenable to a Miller-style pairing algorithm; details are in [Sta24, Algorithm 5.7]. □

To use the pairings  $\widehat{T}_\alpha^\tau$ , the most expedient computation method is the formulas given in Theorem 4 items (5) and (6). In particular, in our applications of  $\widehat{T}_\alpha^\tau$  to form a discrete logarithm problem, in most use cases it suffices to compute  $\widehat{T}_\alpha^\tau(P, Q)^{\bar{\alpha}}$  instead. But if one wishes, one can compute  $\bar{\alpha}^{-1} \pmod{\alpha}$  (provided  $\alpha$  and  $\bar{\alpha}$  are coprime), and use

$$\widehat{T}_\alpha^\tau(P, Q) = \widehat{T}_{N(\alpha)}^\tau(P, Q)^{\bar{\alpha}^{-1}} \pmod{((\mathbb{F}^*)^{\times 2})^\alpha}.$$

This may not apply when  $\alpha$  divides the discriminant.

**Definition 3.** *Also for cryptographic applications, it is convenient to apply a final exponentiation to obtain a reduced pairing, as is common with the classical Tate pairing. This will move the pairing into the roots of unity:*

$$(\overline{\mathbb{F}}^*)/(\overline{\mathbb{F}}^*)^\alpha \rightarrow \mu_{N(\mathcal{O})}^{\times 2} \subseteq (\overline{\mathbb{F}}^*)^{\times 2}, \quad x \mapsto x^{(q-1)\alpha^{-1}}.$$

**Lemma 1.** *Consider the image  $N(\mathcal{O})$  of  $\mathcal{O}$  under the norm map. Then  $N(\mathcal{O})$  modulo  $m > 2$  is a subset of  $\{x^2 : x \in \mathbb{Z}/m\mathbb{Z}\}$  only if  $m$  and  $\Delta_{\mathcal{O}}$  share a non-trivial factor.*

*Proof.* We may assume, by Sunzi’s Theorem (Chinese Remainder Theorem), that  $m$  is a prime power  $p^k > 2$ . If  $p$  is split, the statement follows from the fact that the norm map from  $\mathbb{Q}_p \otimes_{\mathbb{Z}} \mathcal{O}$  to  $\mathbb{Q}_p$  is surjective. If  $p$  is inert, the norm map is surjective on the residue field, so  $N(\mathcal{O})$  modulo  $p^k$  does include non-squares.  $\square$

**Theorem 6.** *Let  $E$  be an elliptic curve oriented by  $\mathcal{O} = \mathbb{Z}[\tau]$ . Let  $m$  be coprime to the discriminant  $\Delta_{\mathcal{O}}$ . Let  $\mathbb{F}$  be a finite field containing the  $m$ -th roots of unity. Suppose  $E[m] = E[m](\mathbb{F})$ . Let  $P$  have order  $m$ . Let  $s$  be the maximal divisor of  $m$  such that  $E[s] \subseteq \mathcal{O}P$ . Then the multiplicative order  $m'$  of  $\widehat{T}_m^\tau(P, P)$  satisfies  $s \mid m' \mid 2s^2$ . In particular, if  $\mathcal{O}P = E[m]$ , then  $s = m$  and the self-pairing has order  $m$ . If  $\mathcal{O}P = \mathbb{Z}P$ , then  $s = 1$ , and in fact, in this case, the self-pairing is trivial.*

To rephrase the last sentence, the self-pairing is trivial on the eigenspaces for the action of  $\mathcal{O}$  on  $E[m]$ . This observation by itself is a consequence of the classification of self-pairings in [CHM+23].

*Proof.* Let  $m' \mid m$  be the order of  $\widehat{T}_m^\tau(P, P)$ . Suppose  $s$  is the maximal divisor of  $m$  so that  $E[s] \subseteq \mathcal{O}P$ . In other words,  $\mathcal{O}P \cong \mathbb{Z}/s\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$  and  $\mathcal{O}P/\mathbb{Z}P \cong \mathbb{Z}/s\mathbb{Z}$  as abelian groups. In particular,  $[s]\mathcal{O}P \in \mathbb{Z}P$ . Thus  $\mathcal{O}[s]P = \mathbb{Z}[s]P$ .

We will show that  $m' \mid 2s^2$  and  $s \mid m'$ . Let  $\lambda \in \mathcal{O}$ . Then  $[\lambda s]P = [ks]P$  for some  $k = k(\lambda) \in \mathbb{Z}$ , and then

$$\widehat{T}_m^\tau([s]P, [s]P)^{k^2} = \widehat{T}_m^\tau([ks]P, [ks]P) = \widehat{T}_m^\tau([\lambda s]P, [\lambda s]P) = \widehat{T}_m^\tau([s]P, [s]P)^{N(\lambda)}.$$

Ranging over all  $\lambda \in \mathcal{O}$ , we conclude that  $N(\lambda)$  are squares modulo  $m'' := m' / \gcd(m', s^2)$ , the multiplicative order of  $\widehat{T}_m^\tau([s]P, [s]P)$ , contradicting that  $m$  is coprime to the discriminant unless  $m'' = 1$  or  $2$  by Lemma 1. Therefore  $m' \mid 2s^2$ . In the case where  $s = 1$ , this argument implies only that the order of  $\widehat{T}_m^\tau(P, P)$  is at most 2. However, the fact that in this case the order of the self-pairing is trivial follows immediately from [CHM+23, Proposition 4.8].

On the other hand, by Theorem 4 item (4), there exists some  $Q$  so that  $\widehat{T}_m^\tau(P, Q)$  has order  $m$ . Let  $t = m/s$ . Then there is a basis for  $E[m]$  of the form  $P, P'$  where  $[t]P' = [\lambda]P$  for some  $\lambda \in \mathcal{O}$ . Writing  $Q = [a]P + [b]P'$ ,

$$\widehat{T}_m^\tau(P, Q)^t = \widehat{T}_m^\tau(P, [t]([a]P + [b]P')) = \widehat{T}_m^\tau(P, [ta + b\lambda]P) = \widehat{T}_m^\tau(P, P)^{ta + b\lambda}.$$

This has order  $s$  on the left. Therefore  $\widehat{T}_m^\tau(P, P)$  must have order a multiple of  $s$ . Hence  $s \mid m'$ .  $\square$

*Remark 2.* As discussed in Sect. 2.3, the authors of [CHM+23], the authors show that non-trivial cyclic self-pairings can only exist for  $P$  of order  $m$  dividing  $\Delta_{\mathcal{O}}$ . The reason our pairings are not ruled out by this result is that our pairings are defined not only on *cyclic subgroups stabilized by the orientation* (where they are in fact trivial, as required).

The following is a partial converse to Theorem 5.

**Theorem 7.** *Let  $E$  be an elliptic curve defined over a finite field  $\mathbb{F}$ , and let  $m \in \mathbb{Z}$ . Let a basis for  $E[m]$  be given. Suppose arithmetic in  $\mathbb{F}$ , discrete logarithms in  $\mathbb{F}^*$  modulo  $m$ , and group law computations on  $E[m]$  can all be accomplished in polynomial time. Suppose  $\varphi(m) > \sqrt{2/3}m$ . Suppose  $E$  is known to be oriented by  $\mathcal{O} = \mathbb{Z}[\tau]$  (but the orientation  $\iota$  is not given), and suppose  $m$  is coprime to the discriminant  $\Delta_{\mathcal{O}}$ . Then the computation of arbitrary pairings  $\widehat{T}_m^\tau(P, Q)$  on  $E[m]$  is Monte-Carlo equivalent in polynomial time to the computation of the action of  $[\tau]$  on  $E[m]$ .*

By Monte-Carlo equivalent, we mean that there is an arbitrarily small probability that the algorithm will return incorrectly. The condition on  $\varphi(m)$  can be improved: what should be required is that  $\varphi(m)$  non-negligibly exceed  $m/\sqrt{2}$ .

*Proof.* Note that computation of  $[\tau]$  allows for computation of  $[\bar{\tau}] = [Tr(\tau)] - [\tau]$ .

If  $[\tau]$  is computable, then by Theorem 4 (5) one can compute  $\widehat{T}_m^\tau(P, Q)$  by computing 3 multiplications by  $[\tau]$  or  $[\bar{\tau}]$ , one addition, and 3 classical Tate pairings.

Conversely, suppose one can compute  $\widehat{T}_m^\tau(P, Q)$  for any  $P, Q \in E[m]$ . We will show how to compute the action of  $[\tau]$  on  $E[m]$ . The pairing is non-degenerate as a consequence of the given hypotheses. It is possible to sample randomly from the subset of order  $m$  points in  $E[m]$ , by choosing  $P$  uniformly randomly as a linear combination  $aP_1 + bP_2$  of the given basis  $P_1, P_2$  such that  $\gcd(a, b)$  is coprime to  $m$ . Choose such a  $P$  of order  $m$  and compute  $\widehat{T}_m^\tau(P, P)$ .

For now, we assume that  $\mathcal{O}P = E[m]$ . Choose  $Q \in E[m]$  so that  $P, Q$  form a basis for  $E[m]$ . Then  $Q = [\lambda]P$  for some  $\lambda \notin \mathbb{Z}$ ; then

$$\widehat{T}_m^\tau(P, Q) = \widehat{T}_m^\tau(P, P)^\lambda.$$

Since  $\widehat{T}_m^\tau(P, P)$  is of order  $m$  by Theorem 6, we can compute  $\lambda$  modulo  $m$  by two pairing computations and a discrete logarithm in  $(\mathbb{F}^*)^{\times 2} / ((\mathbb{F}^*)^{\times 2})^m \cong \mu_m$ .

By construction, we can write  $\tau = a + b\lambda$  modulo  $m$ , so we can compute  $[\tau]P = [a]P + [b]Q$ .

To compute  $[\tau]R$  for arbitrary  $R$ , we first determine  $\mu \in \mathcal{O}$  modulo  $m$  such that  $R = [\mu]P$  (we may use the same discrete log method as above), and then we have  $[\tau]R = [\mu][\tau]P$ .

If  $\mathcal{O}P \neq E[m]$ , then the algorithm is not guaranteed to be correct. Therefore, we run the algorithm several times using different random  $P$  of order  $m$ . We have  $E[m] \cong \mathcal{O}/m\mathcal{O}$  by Theorem 3. Any element of  $\mathcal{O}$  is a  $\mathcal{O}$ -module generator of  $\mathcal{O}/m\mathcal{O}$  provided it is coprime to  $m$  (since 1 is a generator and it has an inverse

modulo  $m$ ). So the proportion of such generators is at least  $(\varphi(m)/m)^2$ . By our assumption on  $m$ , this exceeds  $2/3$ . Any such  $P$  has self-pairing of order  $m$  (by non-degeneracy), so repeating sufficiently often and taking the majority rule answer, this will succeed with overwhelming probability in polynomial time.  $\square$

*Remark 3.* If  $\varphi(m)/m$  is non-negligible, then one can sample points uniformly at random and use the pairing to check whether they generate  $\mathcal{O}$ , with a high probability of success. However, if  $m$  is badly behaved, for example, a primorial, then  $\varphi(m)/m$  may be less than  $1/m^x$  for some  $x > 0$ .

*Remark 4.* Given any basis for  $E[m]$ , the pairing  $\widehat{T}_m^\tau$  allows us to compute the ‘eigenspaces’, i.e. a basis  $P, Q$  such that  $[\tau]P \in \mathbb{Z}P$  and  $[\tau]Q \in \mathbb{Z}Q$ . That is, knowing the pairing values on the original basis, we can solve for points with trivial self-pairing.

*Example 2.* Consider the elliptic curve  $y^2 = x^3 + x$  over  $\mathbb{F}_p$ ,  $p = 541$ . A basis for  $E[5]$  is  $P = (109, 208)$ ,  $Q = (53, 195)$ . If we compute the self-pairings  $\widehat{T}_5^{[i]}([a]P + [b]Q, [a]P + [b]Q)$ , for  $a, b = 0, \dots, 4$ , we obtain the following: the left matrix shows the real parts and the right matrix the imaginary parts, taken to the log base 48 (48 is a generator of  $\mathbb{F}_{541}^*$ ). So, for example, the fourth entry ( $a = 3$ ) in the second column ( $b = 1$ ) (of both matrices) indicates that  $\widehat{T}_5^{[i]}([3]P + [1]Q, [3]P + [1]Q) = (g^3, g^4)$ .

$$\begin{pmatrix} 0 & 4 & 1 & 1 & 4 \\ 0 & 2 & 2 & 0 & 1 \\ 0 & 0 & 3 & 4 & 3 \\ 0 & 3 & 4 & 3 & 0 \\ 0 & 1 & 0 & 2 & 2 \end{pmatrix}, \quad \begin{pmatrix} 0 & 2 & 3 & 3 & 2 \\ 0 & 1 & 1 & 0 & 3 \\ 0 & 0 & 4 & 2 & 4 \\ 0 & 4 & 2 & 4 & 0 \\ 0 & 3 & 0 & 1 & 1 \end{pmatrix}.$$

We can also read off, for example, that  $\widehat{T}_5^{[i]}(P, P) = \widehat{T}_5^{[i]}([2]P + Q, [2]P + Q) = 1$ . Thus the matrices have zeroes on the first column and on the coordinates  $(a, b)$  which are multiples of  $(2, 1)$  modulo 5. This is as dictated by Theorem 6, because  $P \in E[2 + i]$  and  $[2]P + Q \in E[2 - i]$ , which implies  $[i]P = [3]P$  and  $[i]([2]P + Q) = [2]([2]P + Q)$ . In other words, the subgroups  $E[2 \pm i] \cong \mathcal{O}/(2 \pm i)\mathcal{O}$  are the eigenspaces for the action of  $\mathcal{O}$  on  $E[5] \cong \mathcal{O}/5\mathcal{O}$ .

## 5 Recovering Partial Torsion Image Information

Our first observation is that when  $E[m]$  is a cyclic  $\mathcal{O}$ -module, the pairings recover *partial* information about the action of a hidden oriented isogeny  $\phi$  on  $E[m]$ .

**Theorem 8.** *Let  $E$  and  $E'$  be  $\mathcal{O}$ -oriented supersingular curves over  $\overline{\mathbb{F}}_p$  upon which we can efficiently compute the action of a generator  $\tau$  for  $\mathcal{O}$ . Assume that the discrete logarithm in  $\mu_m$  is efficiently computable. Assume also that  $E[m]$  is a cyclic  $\mathcal{O}$ -module, and that the hidden oriented isogeny  $\phi : E \rightarrow E'$  has known degree coprime to  $m$ . Suppose we are given  $P$  and  $P'$  such that  $\mathcal{O}P = E[m]$  and  $\mathcal{O}P' = E'[m]$ . Then we can efficiently recover  $N(\lambda)$  modulo  $m$  for  $\lambda \in \mathcal{O}$  such that  $\phi P = [\lambda]P'$ .*



*Proof.* We have

$$\widehat{T}_m^\tau(P, P)^{\deg \phi} = \widehat{T}_m^\tau(\phi P, \phi P) = \widehat{T}_m^\tau(\lambda P', \lambda P') = \widehat{T}_m^\tau(P', P')^{N(\lambda)}.$$

Note that by Theorem 6,  $\widehat{T}_m^\tau(P, P)$  has order  $m$ . Using the reduced pairing, we can solve a discrete log problem in  $\mu_m$  to obtain  $N(\lambda)$  modulo  $m$ .  $\square$

*Remark 5.* This result improves upon a naïve exhaustive search over the possible images of a general point  $P$  (on account of Theorem 6, we cannot use an eigenvector). More precisely, one could attack the class group action problem by trying all possible image points  $\phi P$  for  $P$ , infer  $\phi[\tau]P = [\tau]\phi P$ , and use the imputed image of  $E[m]$  for the SIDH attacks, checking for success at each attempt. This is similar to [FMP23, Section 4.1], for example. Here, the knowledge of  $N(\lambda)$  restricts  $\phi P$  to typically around  $m$  possible images (between  $m \prod_{\text{prime } q|m} (1 - 1/q)$  and  $m \prod_{\text{prime } q|m} (1 + 1/q)$ ), rather than all  $m^2$ . To run such an attack, we need the degree of  $\phi$  to be known and  $m^2 > \deg \phi$ ,  $m$  to be coprime to  $\deg \phi$ , and  $m$  to be smooth. Since we have great freedom in choosing  $m$ , we can expect to choose an  $m$  around  $\sqrt{\deg \phi}$ .

The example of [CHM+23] described by (1) in Sect. 2.3 shows that when  $m$  is a power of a prime  $\ell$  that splits in  $\mathbb{Q}(\sqrt{-p})$ , the classical  $m$ -Weil pairing also provides an attack with this runtime. However, with the sesquilinear pairing, one does not require splitting conditions.

*Remark 6.* This and other similar results in this paper and in [FFP24] are a caution against Decisional Diffie-Hellman problems in which one must decide if a given point is the image point of a specified torsion point under a hidden isogeny. A result like the previous one reduces the possibilities for the torsion image (without pinning it down entirely). For an example, the IND-CPA hardness of SiGamal [MOT20] depends upon such a problem, called the P-CSSDDH assumption. This is discussed in [CHM+23, Section 6.1], where the authors lament the triviality of the available self-pairings. There are non-trivial pairings of the type  $\widehat{T}$  which would apply to the SiGamal situation, but only if we had access to a different orientation on the curves and isogeny. The Frobenius orientation used in the P-CSSDDH assumption results in a trivial pairing once again, because the torsion is contained in the base field.

*Remark 7.* There is a sense in which we cannot hope to obtain more information than  $N(\lambda)$  modulo  $m$  using these methods. If we post-compose our isogeny with an endomorphism from  $\mathcal{O}$  of norm 1 modulo  $m$ , then we do not change the degree modulo  $m$ , but we do change  $\lambda$ , replacing it with another  $\lambda'$  having the same norm modulo  $m$ . To detect the difference, we must feed in more information than just the degree modulo  $m$ . In fact, it is possible to recover the same result by a different method. Take a basis for  $E$  and  $E'$  and change basis so that the Weil pairing takes a canonical diagonal form. Then the set of possible endomorphisms in  $\mathcal{O}$  that preserve this diagonal form turns out to be the same ‘degree of freedom’ of  $\lambda$  observed above. The pairings from [CHM+23] can be seen as getting around this by assuming  $\lambda \in \mathbb{Z}$ , in which case  $N(\lambda)$  pins down  $\lambda$  more effectively.

*Remark 8.* In principle, the result above doesn't require using  $\widehat{T}$ ; it could be phrased in terms of one of the coordinates in Theorem 4 (5). This wouldn't violate the classification of cyclic self-pairings in [CHM+23] because the domain is not  $\mathbb{Z}$ -cyclic.

## 6 Reduction from $SIDH_1$ to $SIDH$

In [FFP24], the authors consider a variety of variants on the  $SIDH$  problem which can be parameterized by level structure for on the  $m$ -torsion preserved by  $\phi$ . In particular, they define the following problem.

*Problem 1 ( $SIDH_1$ ).* Fix  $d, m \in \mathbb{Z}$ . Let  $E, E'$  be elliptic curves defined over  $\mathbb{F}_q$ , where  $m$  is coprime to  $q$ . Let  $P \in E[m]$  have order  $m$ . Suppose there exists an isogeny  $\phi : E \rightarrow E'$  of known degree  $d$  and  $\phi P$  is given. Find  $\phi$ .

This can be compared to the classical  $SIDH$  problem, in which we are given full torsion image information.

*Problem 2 ( $SIDH$ ).* Fix  $d, m \in \mathbb{Z}$ . Let  $E, E'$  be elliptic curves defined over  $\mathbb{F}_q$ , where  $m$  is coprime to  $q$ . Let  $P, Q$  form a basis for  $E[m]$ . Suppose there exists an isogeny  $\phi : E \rightarrow E'$  of known degree  $d$  and  $\phi P$  and  $\phi Q$  are given. Find  $\phi$ .

In either case we refer to  $m$  as the *level* of the  $SIDH$  or  $SIDH_1$  problem. The authors of [FFP24] show that if  $m$  has a large smooth square factor, then  $SIDH_1$  of level  $m$  (a single torsion point image of order  $m$ ) reduces to  $SIDH$  of level  $O(\sqrt{m})$  (two torsion point images of order  $O(\sqrt{m})$ ). More recently, a manuscript in preparation (presented at Caipi Symposium 2024 [CDM+24]) generalizes the  $SIDH$  attacks of [CD23, MMP+23, Rob23b], directly attacking  $SIDH_1$  without the requirement that  $m$  have a large square factor. Both approaches require that  $m > \deg \phi$ .

Here we show that, *if we have an oriented isogeny*, knowing a single image of order  $m$  is enough to reduce to  $SIDH$  of level  $m$  (on the same curve), assuming only that  $m$  is smooth, with no assumption on  $m$  being square, and no loss in level. Thus using the  $SIDH$  attacks requires only  $m^2 > \deg \phi$ .

Although the proof relies on taking an 'imaginary quadratic viewpoint,' it does not make use of the sesquilinear pairings.

**Theorem 9.** *Let  $E$  and  $E'$  be  $\mathcal{O}$ -oriented supersingular curves over  $\overline{\mathbb{F}}_p$ , upon which we can efficiently compute the action of endomorphisms from  $\mathcal{O}$ . Assume that  $m$  is smooth and coprime to the discriminant. Assume also that  $E[m]$  is a cyclic  $\mathcal{O}$ -module, and that the hidden isogeny  $\phi : E \rightarrow E'$  has known degree coprime to  $m$  and is compatible with the  $\mathcal{O}$ -orientations. Then the problem  $SIDH_1$  of level  $m$  to find  $\phi$  reduces, in a polynomial number of operations in the field of definition of  $E[m]$ , to  $SIDH$  of level  $m$  on the same curve  $E$  and same  $\phi$ .*

*Proof.* For convenience, write  $\mathcal{O} = \mathbb{Z}[\tau]$ . We are given  $\phi R$  for some point  $R \in E[m]$  of order  $m$ . We wish to recover a second torsion point image resulting in an  $SIDH$  problem. First, by Sunzi's Theorem, we can reduce the problem to prime

powers  $m = q^k$ . By assumption,  $q$  is not ramified. Hence we may assume  $q$  is split or inert.

**Case that  $q$  is Inert.** We know  $\mathcal{O}R$  is an  $\mathcal{O}$ -submodule of  $E[m] \cong \mathcal{O}/m\mathcal{O}$ . If  $q$  is inert, it must be isomorphic to  $\mathcal{O}/q^s\mathcal{O}$ . However,  $\mathcal{O}/q^s\mathcal{O}$  doesn't have elements of additive order  $q^k$  unless  $s = k$ . Thus  $\mathcal{O}R = E[m]$ . Given any other point  $Q$ , we may compute  $\eta$  such that  $Q = [\eta]R$  (using basis  $R$  and  $[\tau]R$ ). Then  $\phi Q = \phi[\eta]R = [\eta]\phi R$ .

**Case that  $q$  is Split.** Write  $m = q^k = \mathfrak{b}\bar{\mathfrak{b}}$ , where  $N(\mathfrak{b}) = m$ . Write  $\ker \mathfrak{b} := \{P \in E[m] : \beta P = \mathcal{O} \text{ for all } \beta \in \mathfrak{b}\}$ , and similarly for  $\bar{\mathfrak{b}}$ . Then these are distinct cyclic subgroups of order  $m$ . Thus there exists a basis  $S, T$  for  $E[m]$  so that  $T \in \ker \mathfrak{b}$  and  $S \in \ker \bar{\mathfrak{b}}$ . Similarly, let  $S'$  and  $T'$  be a basis for  $E'[m]$  so that  $T' \in \ker \mathfrak{b}$  and  $S' \in \ker \bar{\mathfrak{b}}$ . To find such subgroups, one can use linear algebra, as follows. The problem of finding  $\ker \mathfrak{b}$  can be rephrased as solving for coefficients  $a$  and  $b$  for  $T = aP + bQ$  in terms of a basis  $P, Q$  for  $E[m]$ , subject to linear conditions determined by the action of  $\mathfrak{b}$ , which we can make explicit in terms of the known action of  $\tau$ . In addition, by adding a gcd condition on the coefficients, one can choose  $T$  to be of full order  $m$ .

Now the mapping  $\phi$ , as a matrix from basis  $S, T$  to basis  $S', T'$ , is diagonal, with some integers  $k_1$  and  $k_2$  on the diagonal (as  $\phi$  respects the  $\mathcal{O}$ -orientation). By writing  $R$  and  $\phi R$  in the relevant bases, namely  $R = [a]S + [b]T$ ,  $\phi R = [c]S' + [d]T'$ , we learn that  $ak_1 \equiv c, bk_2 \equiv d \pmod{m}$ , where  $a, b, c, d$  are known. We also know that  $\deg \phi \equiv k_1 k_2 \pmod{m}$ . Without loss of generality, at least one of  $a$  or  $b$  is coprime to  $m$ , so we know at least one of  $k_1$  or  $k_2$ , and the degree equation then gives us the other. □

## 7 Diagonal SIDH

The following problem arises in [FFP24, Lemma 6 and Sect. 5.6].

*Problem 3 (Diagonal SIDH).* Fix  $d, m \in \mathbb{Z}$ . Let  $E, E'$  be elliptic curves defined over  $\mathbb{F}_q$ , where  $m$  is coprime to  $q$ . Let  $P, Q \in E[m]$  form a basis. Suppose there exists an isogeny  $\phi : E \rightarrow E'$  of known degree  $d$ . Suppose that generators  $P'$  of  $\langle \phi P \rangle$  and  $Q'$  of  $\langle \phi Q \rangle$  are known. Find  $\phi$ .

Interestingly, when the curves are oriented, the Diagonal SIDH problem is amenable to a pairing-based attack, at least for certain conditions on  $E[m]$ .

**Theorem 10.** *Suppose  $E$  and  $E'$  are  $\mathcal{O}$ -oriented (and one can compute the action of the endomorphisms efficiently, as usual). Let  $m > 4 \deg \phi$  be a smooth integer such that modulo  $m$ , 1 has polynomially many square roots. Then Diagonal SIDH with known degree for an oriented isogeny  $\phi : E \rightarrow E'$  is solvable in polynomial time, provided  $\mathcal{O}P = E[m]$  or  $\mathcal{O}Q = E[m]$ .*

*Proof.* Let the Diagonal SIDH problem be given in terms of basis  $P, Q$  for  $E[m]$  and generators  $P'$  and  $Q'$  for  $\langle \phi P \rangle$  and  $\langle \phi Q \rangle$  respectively. Assume without loss of

generality that  $\mathcal{O}P = E[m]$ . Then by Theorem 8, we can efficiently recover  $N(\lambda)$  modulo  $m$  such that  $\phi P = [\lambda]P'$ . However, the Diagonal SIDH setup guarantees that  $\lambda \in \mathbb{Z}$ , hence we have recovered  $\lambda^2$  modulo  $m$ . By assumption, this gives only polynomially many possible values for  $\lambda$ , each of which can be tested by running the SIDH attacks, until one recovers  $\phi$ .  $\square$

An instance of the Diagonal SIDH problem is the problem underlying the FESTA cryptosystem [BMP23, Problem 7]. In this case  $m$  is chosen to be a power of 2, so the attack above would apply if FESTA were instantiated in a situation where the isogeny was oriented (for known orientations). Assuming an  $\mathcal{O}$ -orientation, the condition  $\mathcal{O}P = E[m]$  or  $\mathcal{O}Q = E[m]$  is reasonably likely to occur by chance if not explicitly avoided.

**Corollary 1.** *The hard problem underlying FESTA, namely CIST (see [BMP23]), with  $m > 4 \deg \phi$ , reduces to finding explicit  $\mathcal{O}$ -orientations of the curves  $E$  and  $E'$  respected by the isogeny  $\phi$ .*

*Remark 9.* In [FFP24, Section 5.6], it is shown how to reformulate the problem of finding an isogeny of fixed degree  $d$  between oriented curves (the class group action problem) as a Diagonal SIDH problem, where  $m$  is a product of primes split in  $\mathcal{O}$ . The method of reduction, in brief, uses the eigenspaces associated to a split prime in the orientation, which must map to each other. However, the conditions under which Theorem 10 applies – that  $m$  have few square roots, and  $P$  or  $Q$  be generators of  $E[m]$  as an  $\mathcal{O}$ -module – both fail in the Diagonal SIDH problems that result from the reduction of [FFP24]. This means we cannot chain these attacks together to attack class group action problems!

## 8 When $m$ Divides the Discriminant

Suppose  $m \mid \Delta_{\mathcal{O}}$ , where  $m = N(\tau)$  for  $\tau \in \mathcal{O}$ . In this case the pairing  $\widehat{T}_m^\tau$  becomes trivial. However, a modification is more interesting. Let  $m \in \mathbb{Z}$ , and define

$$T'_m : E[m](\mathbb{F}) \times E(\mathbb{F}) / [m]E(\mathbb{F}) \rightarrow ((\mathbb{F}^*) / (\mathbb{F}^*)^m)^{\times 2},$$

$$T'_m(P, Q) = (t_m([\tau]P, Q), t_m(P, Q)).$$

This modification does not preserve all of the properties of Theorem 4 but importantly, it is bilinear and inherits compatibility from  $t_m$ , so that for  $\phi : E \rightarrow E'$  compatible with  $\mathcal{O}$ , we have

$$T'_m(\phi Q, \phi Q) = T'_m(Q, Q)^{\deg \phi}.$$

In [CHM+23], the authors use generalized pairings to determine the image of a single torsion point in  $E[m]$ , and then reduce to SIDH with  $E[\sqrt{m}]$  when  $m$  is a smooth square. As previously mentioned, recent further development of the SIDH attacks (in preparation [CDM+24]) generalize to image information on

subgroups of a large enough size, not just full torsion subgroups, which effectively removes the restriction that  $m$  be square.

Inspired by this result, we develop a similar reduction using the pairing above. The main advantage of our situation over that in [CHM+23] is the computation of the pairing, which requires only operations in the field of definition of  $E[m]$ . Because the pairings used in [CHM+23] may require a move to the field of definition of  $E[m^2]$ , our pairings result in a speedup in cases where that field of definition is large.

**Proposition 2.** *Let  $E$  be an elliptic curve oriented by  $\mathcal{O} = \mathbb{Z}[\tau]$ . Let  $\mathbb{F}$  be a finite field containing the  $m$ -th roots of unity. Suppose  $E[m] = E[m](\mathbb{F})$  is a cyclic  $\mathcal{O}$ -module. Let  $P \in E[m]$  have order  $m$ . Then the multiplicative order  $T'_m(P, P)$  is at least  $m/t$  where  $t$  is the minimal positive integer such that  $[t]E[m] \subseteq \mathcal{O}P$ .*

*Proof.* The classical Tate-Lichtenbaum pairing

$$t_m : E[m](\mathbb{F}) \times E(\mathbb{F})/[m]E(\mathbb{F}) \rightarrow \mathbb{F}^*/(\mathbb{F}^*)^m$$

is non-degenerate and, for  $P$  of order  $m$ , there exists a  $Q$  so  $t_m(P, Q)$  has order  $m$  (Proposition 1). Let  $t$  be the minimal positive integer for which  $[t]E[m] \subseteq \mathcal{O}P$ . Then  $[t]Q = [a + \tau b]P$ . Using Proposition 1,

$$\begin{aligned} T'_m(P, Q)^t &= T'_m(P, [a + \tau b]P) \\ &= T'_m(P, P)^a T'_m(P, [\tau]P)^b \\ &= (t_m([\tau]P, P)^a t_m([\tau]P, [\tau]P)^b, t_m(P, P)^a t_m(P, [\tau]P)^b) \\ &= \left( t_m([\tau]P, P)^a t_m(P, P)^{N(\tau)^b}, t_m(P, P)^{a + Tr(\tau)b} t_m([\tau]P, P)^{-b} \right). \end{aligned}$$

The left side has order  $m/t$  by Proposition 1. Thus the right side has order  $m/t$ . This is the image of  $T'_m(P, P)$  via a linear transformation of determinant  $N(\tau)b^2 + a^2 + Tr(\tau)ab = N(a + \tau b)$ . Therefore  $T'_m(P, P)$  must have order at least  $m/t$ . □

In the following, we assume  $E[m]$  is a cyclic  $\mathcal{O}$ -module. By Theorem 3, it suffices that the  $\mathcal{O}$ -orientation be  $m$ -primitive.

**Theorem 11.** *Let  $E$  and  $E'$  be  $\mathcal{O}$ -oriented elliptic curves. Suppose there exists an oriented isogeny  $\phi : E \rightarrow E'$  of known degree  $d$ . Let  $m$  be smooth, coprime to  $d$ , and chosen so that there are only polynomially many square roots of 1 modulo  $m$ . Suppose  $m \mid \Delta_{\mathcal{O}}$ . Suppose that  $E[m]$  is a cyclic  $\mathcal{O}$ -module. Suppose  $P \in E[m]$  such that  $\mathcal{O}P = E[m]$ , and  $P' \in E'[m]$  such that  $\mathcal{O}P' = E'[m]$ . Then there exists an efficiently computable point  $Q \in E[m]$  of order  $m$  such that a subset  $S \subset E'[m]$  of polynomial size containing  $\phi(Q)$  can be computed in polynomially many operations in the field of definition of  $E[m]$ .*

*Proof.* Choose a point  $P \in E[m]$  such that  $\mathcal{O}P = E[m]$ . Choose a point  $P' \in E'[m]$  such that  $\mathcal{O}P' = E'[m]$ . Then  $T'_m(P, P)$  and  $T'_m(P', P')$  have order  $m$  by Proposition 2. Then

$$T'_m(P, P)^{\deg \phi} = T'_m(\phi P, \phi P) = T'_m(\lambda P', \lambda P') = T'_m(P', P')^{N(\lambda)}.$$

(Note that  $\lambda$  is an endomorphism, so here we use compatibility with isogenies, not sesquilinearity in general.) Using a discrete logarithm, we can compute  $N(\lambda) \pmod m$ .

Observe that the definition of  $T'_m$  actually depends only on  $\tau$  modulo  $m$ , and hence on  $\mathbb{Z}[\tau]$  modulo  $m$ . So we now choose  $\tau$  in a specific way, possibly not generating all of  $\mathcal{O}$  but only generating it modulo  $m$ . In short, we claim the existence of  $\tau \in \mathcal{O}$  with certain properties, namely that

1.  $\mathbb{Z}[\tau] \equiv \mathcal{O}$  modulo  $m$ ;
2.  $Tr(\tau) \equiv N(\tau) \equiv 0 \pmod{m'}$  where  $m' = m/4$  if  $4 \mid m$ ;  $m' = m/2$  if  $m \equiv 2 \pmod 4$ ; and  $m' = m$  otherwise.

The existence of such a  $\tau$  is a consequence of  $m \mid \Delta_{\mathcal{O}}$ , as follows. A generator for  $\mathcal{O}$  is given by  $\sigma = \frac{\Delta + \sqrt{\Delta}}{2}$  having trace  $\Delta$  and norm  $\frac{1}{4}(\Delta - \Delta^2)$ . Then  $\tau = 2\sigma$  already has the required properties if  $m$  is odd. If  $m$  is even, then  $4 \mid \Delta$ , and the norm is divisible by  $m'$ , so  $\tau = \sigma$  suffices. Hence the minimal polynomial of  $\tau$  is  $x^2$  modulo  $m'$ , and  $[\tau^2]E'[m] \subseteq E'[m/m'] \subseteq E'[4]$ .

Write  $\lambda \equiv a + b\tau$  modulo  $m$ . Then  $N(\lambda) \equiv a^2 \pmod{m'}$ . Since the factorization of  $m'$  is known, by assumption, we have an efficiently computable set of polynomial size of possible values of  $a$ . Compute  $[a][\tau]P'$ . For the correct  $a$ , this is the image of  $[\tau]P$  under  $\phi$  up to addition of a 4-torsion point, since

$$\phi[\tau]P = [\lambda][\tau]P' \in [a][\tau]P' + E'[4].$$

Trying all possible values of  $a$ , and setting  $Q = [\tau]P$ , we obtain the set

$$\{[a][\tau]P' : a^2 \equiv N(\lambda) \pmod m\} + E'[4]$$

required by the statement. Observe that  $P, Q$  form a basis for  $E[m]$  by construction, so  $Q$  has order  $m$ . □

For each possible value of  $a$ , we have a putative  $\phi([\tau]P)$ , i.e. the action of  $\phi$  on a single  $m$ -torsion point. The results of [CHM+23, FFP24] can now be applied if  $m$  is a smooth square,  $d$  is powersmooth and  $m > 4d$ , to reduce to SIDH. Alternatively, loosening the restriction that  $m$  is a square will be possible with the new generalizations of SIDH mentioned above [CDM+24].

The following example demonstrates a new growing family of parameters for which solving the class group action problem (with known degree) is polynomial instead of exponential using Theorem 11.

*Example 3.* This is based on an example communicated to the authors by Wouter Castryck. Let  $E : y^2 = x^3 + x$ . Let  $p$  be a prime of the form  $4 \cdot 3^r - 1$  with  $r > 0$ . This curve is supersingular with  $j = 1728$  and endomorphisms  $[i] : (x, y) \mapsto (-x, iy)$  and  $\pi_p : (x, y) \mapsto (x^p, y^p)$ . Let

$$\tau := \frac{i + \pi_p}{2} \in \text{End}(E).$$

Then  $\tau^2 = -\frac{p+1}{4} = -3^r$ , so  $N(\tau) = 3^r$  and  $Tr(\tau) = 0$ . Let  $\mathcal{O} = \mathbb{Z}[\tau]$ , having  $N(\tau) \mid \Delta_{\mathcal{O}}$ . Let  $m = 3^r$ . Then  $m \mid \Delta_{\mathcal{O}}$ .

Since  $\pi_p^2 = [-p]$ ,  $E(\mathbb{F}_{p^2}) \cong (\mathbb{Z}/(p+1)\mathbb{Z})^2 \cong (\mathbb{Z}/4 \cdot 3^r\mathbb{Z})^2$  [Sil09, Ex. 5.16.d]. Therefore  $E[3^r] \subseteq E(\mathbb{F}_{p^2})$ .

Let  $Q$  be an  $\mathcal{O}$ -generator of  $E[3^r]$ . Then by Proposition 2,  $T'_m(Q, Q)$  has order  $3^r$ , and the polynomially many operations to run the attack of Theorem 11 take place in  $\mathbb{F}_{p^2}$ . The SIDH portion of the attack requires that  $4d < m = 3^r$ .

By contrast, using the methods of [CHM+23, Section 6.1], one would need a generalized pairing value for which only methods of computation taking place in the field of definition of  $E[3^{2r}]$  are known, and this field degree grows exponentially with  $r$ . (Specifically, in [CHM+23, Section 5, p. 20], the authors give an estimated runtime for the pairing needed in the attack, noting that their method requires dividing a point by  $m$  and working in the resulting field extension of degree as much as  $O(m^2)$ .) That means that what was an exponential runtime in terms of  $r$  under [CHM+23] becomes a polynomial one using Theorem 11.

### 9 Supersingular Class Group Action in the Presence of Another Orientation

The following theorem shows that, if we have two distinct orientations respected by  $\phi$ , then we can recover the action of  $\phi$ .

Suppose  $\mathcal{O} \subseteq \text{End}(E)$ . We use the notation  $\mathcal{O}^\perp$  for the quadratic order orthogonal to  $\mathcal{O}$  within the endomorphism ring, with respect to the geometry induced by the quaternion norm.

**Theorem 12.** *Let  $E$  and  $E'$  be supersingular elliptic curves for both of which we know orientations by two quadratic orders  $\mathcal{O}$  and  $\mathcal{O}'$  which together generate a rank 4 sub-order of the endomorphism ring. Let  $\phi : E \rightarrow E'$  be an isogeny of known degree  $d$ . Let  $m$  be smooth, coprime to the discriminants of  $\mathcal{O}$  and  $\mathcal{O}'$ , and suppose 1 has only polynomially many square roots modulo  $m$ . Suppose  $\phi$  respects both the  $\mathcal{O}$  and  $\mathcal{O}'$  orientations. Suppose  $\mathcal{O}$ -module generators are known for both  $E[m]$  and  $E'[m]$ . Suppose, finally, that  $\mathcal{O}^\perp$  has elements of norm coprime to  $m$ . Let  $P \in E[m]$ . Then a subset of  $E'[m]$  of polynomial size containing  $\phi P$  can be computed in a polynomial number of operations in the field of definition of  $E[m]$ .*

*Proof.* Suppose  $E[m] = \mathcal{O}P$  and  $E'[m] = \mathcal{O}P'$ .

Let  $\sigma \in \text{End}(E)$  be chosen to have norm coprime to  $m$ . Write  $\lambda^\sigma$  for an element which participates in the equivalence  $\lambda^\sigma \sigma \equiv \sigma \lambda \pmod{m}$ . Suppose  $\lambda^\sigma \in \mathcal{O}$ . Then

$$\begin{aligned} \widehat{T}_m^\tau([\sigma]P, P)^{\deg \phi} &= \widehat{T}_m^\tau(\phi[\sigma]P, \phi P) \\ &= \widehat{T}_m^\tau([\sigma]\phi P, \phi P) \\ &= \widehat{T}_m^\tau([\sigma][\lambda]P', [\lambda]P') \\ &= \widehat{T}_m^\tau([\lambda^\sigma][\sigma]P', [\lambda]P') \\ &= \widehat{T}_m^\tau([\sigma]P', P')^{\overline{\lambda^\sigma \lambda}}. \end{aligned}$$

Since the norm of  $\sigma$  is coprime to  $m$ ,  $\widehat{T}_m^\tau([\sigma]P, P)$  has the same order as  $\widehat{T}_m^\tau(P, P)$ , which is  $m$  by Theorem 6. Thus, we can compute  $\bar{\lambda}^\sigma \lambda$  modulo  $m$  by performing a discrete logarithm in  $\mu_m$ .

We will now apply the above for two specially chosen  $\sigma \in \text{End}(E)$ . Since we can compute the action of  $\mathcal{O}$  and  $\mathcal{O}'$ , we can compute the action of anything they generate. Thus, we choose  $\sigma_1 = 1$  (so  $\lambda^{\sigma_1} = \lambda$ ), and then some  $\sigma_2 \in \mathcal{O}^\perp$ , so  $\lambda^{\sigma_2} = \bar{\lambda}$ . Assuming that  $\mathcal{O}^\perp$  contains elements of norm coprime to  $m$ , from this, we obtain both  $N(\lambda)$  and  $\lambda^2$  modulo  $m$ .

Using  $N(\lambda)$  and  $\lambda^2$ , we can solve for polynomially many possibilities for  $\lambda$  modulo  $m$  (this requires smoothness, so  $m$  can be factored). Then we have obtained  $\phi P$ . From this we can compute any other  $\phi R$  by solving for  $R = [\mu]P$  and observing that  $\phi R = \phi[\mu]P = [\mu]\phi P$ .  $\square$

**Acknowledgements.** The authors are grateful to Wouter Castryck, Steven Galbraith, Marc Houben, Massimo Ostuzzi, Lorenz Panny, James Rickards and Damien Robert for helpful discussions. They are also grateful to the anonymous referees for feedback which much improved the paper. Both authors have been supported by NSF CAREER CNS-1652238 and NSF DMS-2401580 (PI K. E. Stange).

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References

- [ACKE+24] Sarah Arpin, Wouter Castryck, Jonathan Komada Eriksen, Gioella Lorenzon, and Frederick Vercauteren. International workshop on the arithmetic of finite fields. In *Generalized class group actions on oriented elliptic curves with level structure*, 2024.
- [Arp24] Sarah Arpin. Adding level structure to supersingular elliptic curve isogeny graphs. *Journal de Théorie des Nombres de Bourdeaux*, to appear, 2024.
- [BF23] Andrea Basso and Tako Boris Fouotsa. New SIDH countermeasures for a more efficient key exchange. In *Advances in cryptology—ASIACRYPT 2023. Part VIII*, volume 14445 of *Lecture Notes in Comput. Sci.*, pages 208–233. Springer, Singapore, [2023] 2023.
- [BMP23] Andrea Basso, Luciano Maino, and Giacomo Pope. FESTA: fast encryption from supersingular torsion attacks. In *Advances in cryptology—ASIACRYPT 2023. Part VII*, volume 14444 of *Lecture Notes in Comput. Sci.*, pages 98–126. Springer, Singapore, [2023] 2023.
- [CD23] Wouter Castryck and Thomas Decru. An efficient key recovery attack on SIDH. In *Advances in cryptology—EUROCRYPT 2023. Part V*, volume 14008 of *Lecture Notes in Comput. Sci.*, pages 423–447. Springer, Cham, [2023] 2023.
- [CDM+24] Wouter Castryck, Thomas Decru, Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, Damien Robert, and Benjamin Wesolowski. Isogeny interpolation, 2024. Manuscript in preparation, presented at Caipi Symposium, Rennes, April 30, 2024, [https://caipi\\_symposium.pages.math.cnrs.fr/page-web/editions/avr24.html](https://caipi_symposium.pages.math.cnrs.fr/page-web/editions/avr24.html).





- [CHM+23] Wouter Castryck, Marc Houben, Simon-Philipp Merz, Marzio Mula, Sam van Buuren, and Frederik Vercauteren. Weak instances of class group action based cryptography via self-pairings. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 762–792, Cham, 2023. Springer Nature Switzerland.
- [CHVW22] Wouter Castryck, Marc Houben, Frederik Vercauteren, and Benjamin Wesolowski. On the decisional Diffie-Hellman problem for class group actions on oriented elliptic curves. *Res. Number Theory*, 8(4):Paper No. 99, 18, 2022.
- [CK20] Leonardo Colò and David Kohel. Orienting supersingular isogeny graphs. *J. Math. Cryptol.*, 14(1):414–437, 2020.
- [CL24] Giulio Codogni and Guido Lido. Spectral theory of isogeny graphs, 2024. <https://arxiv.org/abs/2308.13913>.
- [CLG09] Denis X. Charles, Kristin E. Lauter, and Eyal Z. Goren. Cryptographic hash functions from expander graphs. *J. Cryptology*, 22(1):93–113, 2009.
- [CLM+18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In *Advances in cryptology—ASIACRYPT 2018. Part III*, volume 11274 of *Lecture Notes in Comput. Sci.*, pages 395–427. Springer, Cham, 2018.
- [Cou06] Jean-Marc Couveignes. Hard homogeneous spaces. Cryptology ePrint Archive, Paper 2006/291, 2006. <https://eprint.iacr.org/2006/291>.
- [Cox22] David A. Cox. *Primes of the Form  $x^2 + ny^2$ : Fermat, Class Field Theory, and Complex Multiplication. Third Edition with Solutions*. AMS Chelsea Publishing, 2022.
- [CSV22] Wouter Castryck, Jana Sotáková, and Frederik Vercauteren. Breaking the decisional Diffie-Hellman problem for class group actions using genus theory: extended version. *J. Cryptology*, 35(4):Paper No. 24, 30, 2022.
- [CVCCD+19] Daniel Cervantes-Vázquez, Mathilde Chenu, Jesús-Javier Chi-Domínguez, Luca De Feo, Francisco Rodríguez-Henríquez, and Benjamin Smith. Stronger and faster side-channel protections for CSIDH. In *Progress in cryptology—LATINCRYPT 2019*, volume 11774 of *Lecture Notes in Comput. Sci.*, pages 173–193. Springer, Cham, 2019.
- [EHL+20] Kirsten Eisenträger, Sean Hallgren, Chris Leonardi, Travis Morrison, and Jennifer Park. Computing endomorphism rings of supersingular elliptic curves and connections to path-finding in isogeny graphs. In *ANTS XIV—Proceedings of the Fourteenth Algorithmic Number Theory Symposium*, volume 4 of *Open Book Ser.*, pages 215–232. Math. Sci. Publ., Berkeley, CA, 2020.
- [FFP24] Luca De Feo, Tako Boris Fouotsa, and Lorenz Panny. Isogeny problems with level structure. Springer-Verlag, 2024.
- [FMP23] Tako Boris Fouotsa, Tomoki Moriya, and Christophe Petit. M-SIDH and MD-SIDH: countering SIDH attacks by masking information. In *Advances in cryptology—EUROCRYPT 2023. Part V*, volume 14008 of *Lecture Notes in Comput. Sci.*, pages 282–309. Springer, Cham, [2023] 2023.
- [GPV23] Steven D. Galbraith, Derek Perrin, and José Felipe Voloch. CSIDH with level structure. Cryptology ePrint Archive, Paper 2023/1726, 2023.
- [IJ13] Sorina Ionica and Antoine Joux. Pairing the volcano. *Math. Comp.*, 82(281):581–603, 2013.

- [KLPT14] David Kohel, Kristin Lauter, Christophe Petit, and Jean-Pierre Tignol. On the quaternion  $\ell$ -isogeny path problem. *LMS J. Comput. Math.*, 2014.
- [KT19] Takeshi Koshihara and Katsuyuki Takashima. New assumptions on isogenous pairing groups with applications to attribute-based encryption. In *Information security and cryptology—ICISC 2018*, volume 11396 of *Lecture Notes in Comput. Sci.*, pages 3–19. Springer, Cham, 2019.
- [Len96] H. W. Lenstra, Jr. Complex multiplication structure of elliptic curves. *J. Number Theory*, 56(2):227–241, 1996.
- [Mil04] Victor S. Miller. The weil pairing, and its efficient calculation. *J. Cryptology*, 17(4):235–261, 2004.
- [MMP+23] Luciano Maino, Chloe Martindale, Lorenz Panny, Giacomo Pope, and Benjamin Wesolowski. A direct key recovery attack on SIDH. In *Advances in cryptology—EUROCRYPT 2023. Part V*, volume 14008 of *Lecture Notes in Comput. Sci.*, pages 448–471. Springer, Cham, [2023] 2023.
- [MOT20] Tomoki Moriya, Hiroshi Onuki, and Tsuyoshi Takagi. SiGamal: a supersingular isogeny-based PKE and its application to a PRF. In *Advances in cryptology—ASIACRYPT 2020. Part II*, volume 12492 of *Lecture Notes in Comput. Sci.*, pages 551–580. Springer, Cham, [2020] 2020.
- [Rei23] Krijn Reijnders. Effective pairings in isogeny-based cryptography. In *Progress in cryptology—LATINCRYPT 2023*, volume 14168 of *Lecture Notes in Comput. Sci.*, pages 109–128. Springer, Cham, [2023] 2023.
- [Rob23a] Damien Robert. The geometric interpretation of the Tate pairing and its applications. Cryptology ePrint Archive, Paper 2023/177, 2023. <https://eprint.iacr.org/2023/177>.
- [Rob23b] Damien Robert. Breaking SIDH in polynomial time. In *Advances in cryptology—EUROCRYPT 2023. Part V*, volume 14008 of *Lecture Notes in Comput. Sci.*, pages 472–503. Springer, Cham, [2023] 2023.
- [RS06] Alexander Rostovtsev and Anton Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Paper 2006/145, 2006. <https://eprint.iacr.org/2006/145>.
- [Sil09] Joseph H. Silverman. *The Arithmetic of Elliptic Curves, Second Edition*. Springer, 2009.
- [Sta24] Katherine E. Stange. Sesquilinear pairings on elliptic curves, 2024. <https://arxiv.org/abs/2405.14167>.
- [Wes22] Benjamin Wesolowski. The supersingular isogeny path and endomorphism ring problems are equivalent. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science—FOCS 2021*, pages 1100–1111. IEEE Computer Soc., Los Alamitos, CA, [2022] 2022.



# SQIPrime: A Dimension 2 Variant of SQISignHD with Non-smooth Challenge Isogenies

Max Duparc<sup>(✉)</sup> and Tako Boris Fouotsa

EPFL, Lausanne, Switzerland  
{max.duparc,tako.fouotsa}@epfl.ch

**Abstract.** We introduce SQIPrime, a post-quantum digital signature scheme based on the Deuring correspondence and Kani’s Lemma. Compared to its predecessors that are SQISign and especially SQISignHD, SQIPrime further expands the use of high dimensional isogenies, already in use in the verification in SQISignHD, to all its subroutines. In doing so, it no longer relies on smooth degree isogenies (of dimension 1). Intriguingly, this includes the challenge isogeny which is also a non-smooth degree isogeny, but has an accessible kernel. The fact that the isogenies do not have rational kernel allows to fit more rational power 2 torsion points which are necessary when computing and representing the response isogeny. SQIPrime operates with prime numbers of the form  $p = 2^\alpha f - 1$ .

We describe two variants of SQIPrime. SQIPrime4D which incorporates the novelties described above and uses dimension 4 isogenies to represent the response isogeny. The runtime of higher dimensional isogeny computation is exponential in the dimension, hence the smaller the dimension the better for efficiency. The second variant, SQIPrime2D, solely uses dimension 2 isogenies. This is achieved by setting the degree of the secret isogeny to be equal to that of the challenge isogeny and further exploiting Kani’s Lemma. SQIPrime2D is more efficient compared to SQIPrime4D and to SQISignHD, at the cost of being comparatively less compact, but still very compact compared to non isogeny based post-quantum signatures.

**Keywords:** Isogenies · SQISign · SQISignHD · Kani’s Lemma · SQIPrime

## 1 Introduction

The interest of isogeny based signature schemes is that they provide compact post-quantum signatures. This property, which comes at the cost of a greater computational cost, motivated their research. Among the early propositions of isogeny based signature schemes such as [6, 16, 49], was GPS [27] that specifically relied on Deuring correspondence [20]. Its ideas were expanded and improved in 2020 by De Feo, Kohel, Leroux, Petit and Wesolowski to create

the SQISign protocol in [18]. As of today, SQISign is the only isogeny based candidate at the NIST [38] post-quantum cryptography standardization effort. In 2023, Dartois, Leroux, Robert and Wesolowski proposed SQISignHD [11], a variant of SQISign utilising Kani’s Lemma [29] for verification. Both SQISign (and follow-ups [19, 46]) and SQISignHD are, as of today, the two most compact post-quantum signatures, of respective size 177B for SQISign and 109B for SQISignHD for 128 bits of security.

Kani’s Lemma and high dimensional isogenies (originally used in [8, 34, 44] to prove that SIDH [17, 28] was insecure by leveraging accessible images of torsion points) are used in SQISignHD to solve some drawbacks of SQISign as they can be used to represent isogenies of non-smooth degree, which significantly simplifies the signature part of SQISignHD, at the cost of a more complex verification. The emergence of SQISignHD is part of a broader trend in Isogeny Based Cryptography, consisting in leveraging the new capabilities enabled by Kani’s Lemma, a trend that birthed many new cryptographic schemes such as SQISignHD [11], FESTA and QFESTA [3, 36], IS-CUBE [35], SCALLOP-HD [9], DeuringVRF [33], SILBE [22] or POKE [1]. Kani’s lemma has also been recently used to design a new ideal-to-isogeny algorithm [39] for the SQISign signature scheme.

As mentioned above, the main input in SQISignHD is the use of high dimensional isogenies to represent the response. In SQISign, the secret key is an isogeny  $\tau : E_0 \rightarrow E_A$ , where  $E_0$  has  $j$ -invariant 1728. The commitment is a curve  $E_1$  obtained by computing an isogeny  $\psi : E_0 \rightarrow E_1$  and the challenge is an isogeny  $\varphi : E_1 \rightarrow E_2$ . The response is an isogeny  $\sigma : E_A \rightarrow E_2$  (see left-hand side of Fig. 1). The isogeny  $\sigma$  is in fact a long smooth isogeny of degree roughly  $p^{15/4}$ , obtained through a more efficient variant [18, 19] of the KLPT algorithm [30]. The use of the KLPT algorithm and the fact that the degree of the response isogeny  $\sigma$  is roughly  $p^{15/4}$  implies that one needs to use primes with as much accessible (defined over a small extension of  $\mathbb{F}_p$ ) smooth torsion as possible. This is one of the biggest constraints in SQISign that was solved in SQISignHD.

The attacks [8, 34, 44] on SIDH/SIKE (and any other isogeny based protocol revealing images of smooth order torsion points such as [10, 14, 25]) led to a new method for representing isogenies of generic degree [43]. In fact, an evaluation of an isogeny on torsion points of large (with respect to the degree of the isogeny) smooth order is a representation of this isogeny. In SQISignHD, from the knowledge of the endomorphism rings of the curves at play, the signer samples a relatively short (but non-smooth) response isogeny  $\sigma$  and evaluates it on torsion points of smooth order. This evaluation is then returned to the verifier as the response. Since this evaluation represents the isogeny, the verifier can efficiently check that the data received represents an isogeny  $\sigma : E_1 \rightarrow E_2$ . Note that here, the response goes from  $E_1$  to  $E_2$  while the challenge goes from  $E_A$  to  $E_2$ , this change is made for a more convenient implementation. This brings several relaxations, among which the change of the base prime  $p$  to an SIDH prime:  $p = 2^a 3^b f - 1$ . In SQISign, the most computationally involved part is transforming the ideal obtained from KLPT into an isogeny, this is done during the signing process. In SQISignHD, signing is somewhat easier since the KLPT algorithm is avoided, but the verification is computationally involved. In fact,

in order to validate that the evaluation returned by the signer represents an isogeny  $\sigma : E_1 \rightarrow E_2$ , one needs to compute and evaluate an isogeny in higher dimension: 2, 4 or 8 in general. The smaller the dimension, the more efficient the computation and the evaluation are. In SQISignHD, the verification uses dimension 4 isogenies. There is a huge efficiency gap between dimension 4 isogenies and dimension 2 isogenies [11, 12, 31, 45]. Hence, in the quest for better efficiency, it becomes natural to ask the following question:

*Can one design a variant of SQISignHD that uses only dimension 1 and/or dimension 2 isogenies?*

**Contributions.** In this paper, we answer the question above in the affirmative, by describing SQIPrime, a derivative of SQISignHD. To do so, we first extend the use of Kani’s Lemma to both key generation and commitment, by adapting the **RandIsogImages** algorithm from QFESTA [36]. Next, we modify the challenge isogeny generation in such a way that the verifier can use non-smooth degree isogenies, by sampling solely the kernel generator of this isogeny. The signer/prover can then use the techniques introduced by Leroux [33] to compute this challenge isogeny and include it in the response. As a consequence, we use primes of the form  $p = 2^\alpha f - 1 = 2Nq + 1$  where  $q$  is the degree of the challenge. These changes induce numerous adaptations and optimizations throughout the protocol. In order to ease understanding and not apply all the numerous changes at once, we propose two variants of SQIPrime: SQIPrime4D and SQIPrime2D.

In SQIPrime4D, we incorporate the most basic changes to SQISignHD, without necessarily aiming for a better efficiency. These changes include: the use of an adaptation (**KaniDoublePath**, Sect. 3.1) of the **RandIsogImages** algorithm from QFESTA [36] for key generation and commitment, and the use of a non-smooth degree isogeny for commitment. More precisely, let  $\tau : E_0 \rightarrow E_A$ ,  $\psi : E_0 \rightarrow E_1$ ,  $\varphi : E_A \rightarrow E_2$  and  $\sigma : E_1 \rightarrow E_2$  be the secret, commitment, challenge and response isogenies in SQISignHD. In SQIPrime4D,  $\tau$  and  $\psi$  are generated using the **KaniDoublePath** algorithm. For the challenge, the verifier samples a uniformly random scalar  $a \in \mathbb{Z}_q$  where  $q$  is the degree of the commitment isogeny. The scalar  $a$  defines a point  $C = P + [a]Q$  where  $(P, Q)$  is a specified basis of  $E_A[q]$ . The signer/prover uses the techniques in the DeuringVRF [33] to translate  $C$  into its corresponding ideal  $I_\varphi$ , which is in fact the ideal corresponding to the challenge isogeny  $\varphi : E_A \rightarrow E_2$ . From here, they recover the endomorphism ring of  $E_2$ , solve for a short isogeny  $\sigma : E_2 \rightarrow E_1$  (note that this is the dual of the response in the original SQISignHD), and evaluate  $\kappa = \sigma \circ \varphi$  on the  $2^\alpha$ -torsion points (this is illustrated in Fig. 2). The evaluation of  $\kappa = \sigma \circ \varphi$  is then returned to the verifier as the response. The verifier checks that the data they received represents an isogeny  $\kappa : E_A \rightarrow E_1$  of degree  $qd$  whose kernel contains  $C = P + [a]Q$  and,  $q$  and  $d$  are co-prime. This proves that  $\kappa$  factors through the challenge  $\varphi : E_A \rightarrow E_2$  whose kernel was sampled by the verifier. The verification is performed using dimension 4 isogenies. In SQIPrime2D, we implement further adjustments in order to use only dimension 2 isogenies.

The main obstacle when representing isogenies in dimension 2 is the need of an auxiliary isogeny. To represent the isogeny  $\kappa := \sigma \circ \varphi : E_A \rightarrow E_1$  of degree  $qd$  returned in SQIPrime4D in dimension 2, we need an auxiliary isogeny  $\delta : E_A \rightarrow E_\delta$  of degree  $2^\alpha - qd$ . Hence, the goal of all the changes we will operate from now on will be to enable an efficient computation of such an auxiliary isogeny. The main change consists in fixing the degree of the secret isogeny  $\tau$  to  $q$ , the same degree as that of the challenge isogeny  $\varphi$ , and making sure that this degree is prime. Once this is done, we sample an endomorphism  $\gamma \in \text{End}(E_0)$  of degree  $d(2^\alpha - dq)$ , and compose it with the secret isogeny  $\tau : E_0 \rightarrow E_A$  to obtain an isogeny  $\tau \circ \gamma : E_0 \rightarrow E_A$  of degree  $dq(2^\alpha - dq)$ . This isogeny can be seen as the composition of two isogenies of degree  $dq$  and  $2^\alpha - dq$  respectively. We then use Kani's Lemma to recover the pushforward of the isogeny of degree  $2^\alpha - dq$  in such a way that its domain is  $E_A$ , and its codomain is some curve  $E_\delta$  which is computed at the same time. This pushforward is used as the sought auxiliary isogeny, allowing us to have a variant SQIPrime2D which only uses dimension 2 isogenies. The SQIPrime2D identification scheme is illustrated in Fig. 3.

The key generation in SQIPrime2D requires two dimension 2 isogeny computation and evaluation. The signing process requires two dimension 2 isogeny computations and evaluations, one for the commitment isogeny and another for generating the auxiliary isogeny. The verification requires one dimension 2 isogeny computation and evaluation, bringing it up to a total of three dimension 2 isogeny computations and evaluations for the signature and verification. Given the current efficiency gap between dimension 2 and dimension 4 isogenies, we expect SQIPrime2D to be more efficient compared to SQISignHD. This is to be confirmed with a more advanced implementation of SQIPrime2D, task that we leave as future work.

In order to prove the security of SQIPrime4D and SQIPrime2D, we assume that the codomain of an isogeny computed using the **KaniDoublePath** algorithm is computationally indistinguishable from a random supersingular curve. Once this assumption is made, we reduce the security of SQIPrime4D and SQIPrime2D to the Supersingular Endomorphism problem in the RUCGDIO or RUCODIO+AIO models respectively, models that we introduce and which are translations of the RUDGIO model (introduced in the context of SQISignHD) into the context of SQIPrime4D and SQIPrime2D respectively.

**Related Work.** While this work was under finalisation, we became aware of two other concurrent but independent projects that were trying to answer the same open question we answer in the paper. The first project is from Nakagawa and Onuki, named SQISign2D-East [37] and the second one is from Basso, Dartois, De Feo, Leroux, Maino, Pope, Robert and Wesolowski, named SQISign2D-West [2]. Interestingly, all three papers adopt substantially different approaches to solve this problem.

- Our mechanism mainly relies on the primality of the challenge isogeny  $\varphi$  and on the fact that it has the same degree as our secret isogeny  $\tau$ .

- The SQISign2D-East [37] mechanism uses Eichler modules [32, Definition 1.2.7] to sample endomorphisms over  $E_0$  that can also be interpreted as endomorphisms over  $E_A$ . The auxiliary isogeny  $\delta : E_A \rightarrow E_\delta$  is then generated using such endomorphisms on  $E_A$ .
- Finally, the SQISign2D-West [2] mechanism merges **RandIsogImages** with Clapoti [40] to design a new efficient algorithm to evaluate random ideals. This algorithm is then used to compute the auxiliary isogeny by sampling its ideal, composing it with the commitment and challenge ideals, evaluating the composition. Using the knowledge of the commitment and challenge isogenies, the auxiliary isogeny is retrieved.

We wholeheartedly recommend the reader to delve into these two papers (after completing ours, naturally).

**Outline.** The remainder of this paper is organised as follows. In Sect. 2, we give a quick recall on the architecture of both SQISign and SQISignHD, together with a reminder of the standard algorithms in Isogeny Based Cryptography that we use to define SQIPrime. In Sect. 3, we will introduce special tools that we will need to construct both SQIPrime4D and SQIPrime2D. In Sect. 4, we give the detailed construction of SQIPrime4D, together with an analysis of its security in Sect. 5. Similarly, we give the detailed specification of SQIPrime2D in Sect. 6, with its security analysis in Sect. 7. Finally, we discuss in Sect. 8 how to find adequate parameters for both SQIPrime4D and SQIPrime2D and have a word about their foreseen efficiency.

## 2 Background

We assume some familiarity with Isogeny Based Cryptography. We provide in [21, Appendix A] a concise overview of isogenies, Deuring correspondence, and Kani’s Lemma. For a more comprehensive exploration, we recommend referring to De Feo’s notes [13] and Silverman’s book [47] for a general understanding of elliptic curves and isogenies. For insights into the Deuring Correspondence, Leroux’s thesis [32] is an excellent resource, while Robert’s attack on SIDH [43, 44] provides valuable details on Kani’s Lemma.

Throughout this paper, we denote by  $\lambda$  the security parameter. Let  $p$  be a prime,  $\mathbb{F}_p$  is the finite field of cardinality  $p$ . We denote as  $E_0$  the curve with  $j$ -invariant 1728 given by  $y^2 = x^3 + x$ . If  $p \equiv 3 \pmod{4}$ , then it is supersingular and its endomorphism ring correspond to the maximal order  $\mathcal{O}_0 = \mathbb{Z} + \mathbf{i}\mathbb{Z} + \frac{1+\mathbf{j}}{2}\mathbb{Z} + \frac{1+\mathbf{i}\mathbf{j}}{2}\mathbb{Z}$  with  $\mathbf{i} : (x, y) \rightarrow (-x, \sqrt{-1}y)$  and  $\mathbf{j} = \pi$  the Frobenius endomorphism. This is an evaluation basis<sup>1</sup> denoted  $\mathfrak{D}_0$ .

---

<sup>1</sup> An evaluation basis [11, Definition A.4.1] consists in an isomorphism between the endomorphism ring and a maximal order such that every element of the basis is efficiently computable.

## 2.1 Standard Algorithms

SQIPrime, even more profoundly than SQISign and SQISignHD, heavily relies on the different efficient representations [11, Definition 1] of isogenies and more specifically the kernel, ideal and high dimensional representations. To do so, it uses the following standard algorithms in Isogeny Based Cryptography:

- **KernelToIsogeny**: Takes as input  $E$  a supersingular curve and  $K \in E[d]$  and returns  $\phi$  the isogeny of degree  $d$  whose kernel is generated by  $K$  together with  $E'$ , its codomain. To do so, it uses Vélú’s Formulas [48] and factorises  $\phi$  as a composition of prime degree isogenies. To be efficient,  $d$  needs to be smooth.<sup>2</sup>
- **CanonicalTorsionBasis**: Takes as input  $E$  a supersingular curve and  $N$  an integer such that  $N|(p^2 - 1)$  and returns  $\langle P, Q \rangle = E[N]$ . To do so, it simply samples points at random in  $E(\mathbb{F}_{p^2})$  or its quadratic twist and multiplies it by the right cofactor. To ensure that this method is deterministic, the sampling is performed deterministically using the Elligator algorithm [5].
- **KernelToIdeal** [11, Algorithm 9]: Takes as input  $\mathfrak{O}_E$  an evaluation basis of  $\text{End}(E)$  and  $K$  a generator of the kernel of an isogeny  $\phi$  of smooth degree  $d$  and returns  $I_\phi$ .
- **FullRepresentInteger** [32, Algorithm 4]: Takes as input a number  $N > 4p$  and returns  $\gamma \in \mathcal{O}_0$  an endomorphism of  $E_0$  of norm  $N$ . Note that the successful termination of this algorithm relies on plausible heuristics. We refer to [32, Section 3.1] for further details.
- **EvalTorsion** [11, Algorithm 11]: It takes as input  $\mathfrak{O}_F$  an evaluation basis of  $\text{End}(F)$ ,  $\rho_1 : F \rightarrow E$  of degree  $d_1$ ,  $\rho_2 : F \rightarrow E'$  of degree  $d_2$ , both efficiently computable isogenies together with their respective ideals  $I_1$  and  $I_2$ . It also takes as input  $J$  an  $(\mathcal{O}_E, \mathcal{O}_{E'})$ -ideal of norm  $N$  co-prime to  $d_1$  and  $d_2$ . It outputs  $\phi_J(P)$ , with  $P$  any point whose order is co-prime to  $d_1 d_2$ .
- **RandomEquivalentIdeal** [32, Algorithm 6]: It takes as input a  $(\mathcal{O}_E, \mathcal{O}_F)$ -ideal  $I$  and returns  $J$  another  $(\mathcal{O}_E, \mathcal{O}_F)$ -ideal such that  $n(J)$  is a “small” prime, meaning that  $n(J) \simeq \sqrt{p}$  with extremely high probability, as shown in [32, Lemma 3.2.3 & Lemma 3.2.4].
- **HDKernelToIsogeny**: This is an high dimensional equivalent to **KernelToIsogeny**. Depending on the dimension, it can be based upon theta structures [11, 12, 42], or over Kummer surfaces [45].

## 2.2 SQISign and SQISignHD

The SQISign and SQISignHD signature algorithms are in fact  $\Sigma$ -protocols that are transformed into digital signature schemes using the Fiat-Shamir transform [23], rendering them Universally Unforgeable under Chosen Message Attacks (UU-CMA) secure in the Random Oracle Model (ROM). The underlying  $\Sigma$ -protocols are built upon the Deuring correspondence, hence the acronym SQIS

<sup>2</sup> Note that this algorithm, as presented here, is not optimal. Among the important improvements on those computations, see [17] and [4].

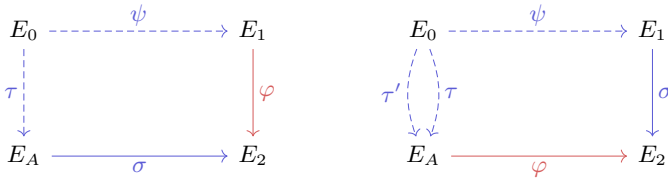


for Short Quaternion Identification Scheme. The security of both protocols relies on the hardness of the *one endomorphism problem* (Problem 1). The one endomorphism problem was recently [41] shown to be equivalent to the endomorphism ring problem (Problem 2), a central problem in isogeny based cryptography, which is believed to be hard for both classical and quantum adversaries.

*Problem 1.* Let  $E$  be a random supersingular curve defined over  $\mathbb{F}_{p^2}$ , find a nontrivial (not in  $\mathbb{Z}$ ) endomorphism of  $E$ .

*Problem 2.* Let  $E$  be a random supersingular curve defined over  $\mathbb{F}_{p^2}$ , compute the endomorphism ring  $\text{End}(E)$  of  $E$ .

The main idea behind SQISign and SQISignHD is to prove the knowledge of the endomorphism ring  $\text{End}(E_A)$  of  $E_A$ , a supersingular curve. In SQISign, the fact that the prover knows  $\text{End}(E_A)$  enables them to find a connecting isogeny between  $E_A$  and any other curve  $E_2$ , provided that they also know  $\text{End}(E_2)$ . The idea is then to let  $E_2$  be chosen as the challenge by the verifier, by computing a random isogeny  $\varphi : E_1 \rightarrow E_2$  where  $E_1$  was generated by the prover (who hence knows its endomorphism ring  $\text{End}(E_1)$ ). Using  $\varphi$ , the prover can retrieve  $\text{End}(E_2)$  and respond with an isogeny  $\sigma : E_A \rightarrow E_2$  that can be easily verified. This is illustrated in Fig. 1. The high level picture in SQISignHD is similar with a minor exception that the domain of  $\varphi$  and  $\sigma$  are interchanged for efficiency reasons. The main difference between SQISign and SQISignHD consists in how the response isogeny  $\sigma$  is computed and represented. The first returns a very long smooth isogeny through a sequence of kernels, while the second uses high dimension isogenies to represent a relatively short but non-smooth isogeny.



**Fig. 1.** Diagrams of SQISign (left) and SQISignHD (right). The prover is in blue and the verifier is in red. Dashed isogenies are secrets. (Color figure online)

**SQISign:** To construct  $\sigma$  the connecting isogeny, SQISign uses a variant of the **KLPT** [30] named the **SigningKLPT** [18, Algorithm 5]. The ideal  $I_\sigma$  it retrieves is smooth, as its norm is a large power of 2 of size  $O(p^{15/4})$ . To be efficiently computed,  $\sigma$  is represented as a composition of isogenies with rational kernel generator. Transcribing  $I_\sigma$  to these kernels is done using **IdealToIsogeny**

[19, Algo. 7]. This **IdealToIsogeny** step requires a lot of smooth torsion, reason why the prime  $p$  is such that  $2^\ell T | p^2 - 1$  with  $T \simeq p^{5/4}$  and  $T$  smooth. Finding such primes is *difficult* and  $T$  often has prime factors in the order of  $10^3$ . Those big factors significantly slow down the signing procedure, as several  $T$  isogenies have to be computed throughout **IdealToIsogeny**. On the other hand, the verification of SQISign is very efficient, as it essentially consists in computing a sequence of isogenies of degree  $2^\ell$  from their kernels.

**SQISignHD**: On the other hand, SQISignHD uses the **RandomEquivalentIdeal** to compute  $\sigma$ . The response isogeny is therefore short  $O(\sqrt{p})$  but not smooth. It is given to the verifier using high dimension representation [43]. This shift to high dimension isogenies considerably speeds up the signature part of SQISignHD but shifts most of the expensive computation to the verification that has to use Kani’s Lemma in dimension 4. To be efficient, SQISignHD uses “SIDH-like” prime, that are easy to find. We refer to [11] for further details.

### 3 Introduced Techniques

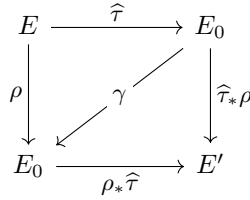
Before jumping into SQIPrime, we detail two new techniques that we will use to construct our variant of SQISignHD.

1. The first tool is called **KaniDoublePath**, a variant of **DoublePath** [11, Section 3.3] that uses Kani’s Lemma to sample two (possibly non-smooth) isogenies between  $E_0$  and  $E_A$  of co-prime degrees. This algorithm is a modification of the **RandIsogImages** [36, Algorithm 2], as it additionally computes the corresponding ideals of these isogenies. We also describe a variant **ExtKaniDoublePath** that relies on endomorphisms of greater norm.
2. The second is a method to compute, given  $K$  a generator of the kernel of an isogeny, the corresponding ideal even when the degree of this isogeny is non-smooth. This method is an adaptation of the work of Leroux on DeuringVRF [33] and allows us to use large non-smooth degree isogenies as challenge isogeny in SQIPrime.

#### 3.1 KaniDoublePath

The main idea behind **KaniDoublePath** is, similarly to the **DoublePath** algorithm, to construct two isogenies of co-prime degree between  $E_0$  and another supersingular curve  $E$ . The main interest of **KaniDoublePath** lies in the fact that those isogenies are not necessary smooth.

To perform the **KaniDoublePath**, we first use **FullRepresentInteger** to find an endomorphism  $\gamma \in \text{End}(E_0)$  with  $\deg(\gamma) = \ell(N - \ell)$  with  $\ell, N$  co-prime and  $N$  smooth. We can decompose  $\gamma$  as  $\gamma = \rho \circ \tau$  with  $\deg \tau = \ell$  and  $\deg \rho = N - \ell$ . Using Kani’s Lemma, we compute the dimension 2 isogeny  $F : E_0 \times E_0 \rightarrow E \times E'$  given by the following diagram and kernel:



$$\ker(F) = \left\{ ([-\ell](P), \gamma(P)) \mid P \in E_0[N] \right\} \text{ with } F := \begin{pmatrix} \tau & -\widehat{\rho} \\ \widehat{\tau}_* \rho & \rho_* \widehat{\tau} \end{pmatrix}$$

We can therefore efficiently evaluate both  $\tau$  and  $\widehat{\rho}$  at any points of  $E_0$  by writing  $\tau(-) = F(-, 0)_1$  and  $\widehat{\rho}(-) = -F(0, -)_1$ . Additionally, we also retrieve  $I_\tau$  and  $I_\rho$  the ideal corresponding to  $\tau$  and  $\rho$  as  $I_\tau = \mathcal{O}_0\gamma + \mathcal{O}_0\ell$  and  $I_\rho = \mathcal{O}_0\overline{\gamma} + \mathcal{O}_0(N - \ell)$ . The full process is summarized in Algorithm 1.

One may ask if a curve generated using **KaniDoublePath** has the same distribution as a curve generated by sampling a random cyclic kernel of size  $\ell$  and computing the corresponding isogeny. In practice, if the degree  $N - \ell$  of the byproduct isogeny  $\rho$  is not way larger than  $p$ , it may happen that for some curve  $E$  which is  $\ell$ -isogenous to  $E_0$ , there exists no isogeny of degree  $N - \ell$  between  $E_0$  and  $E$ , meaning that  $E$  will never be returned by **KaniDoublePath**. We describe **ExtKaniDoublePath**, a variation of **KaniDoublePath** in which the degree of the byproduct isogeny  $\rho$  is larger, hence increasing the chances that there exists such an isogeny between  $E_0$  and any curve which is  $\ell$ -isogenous to  $E_0$ , hence reducing the gap between the two distributions.

---

**Algorithm 1. KaniDoublePath**

---

**Input:**  $\mathfrak{D}_0$  the evaluation basis of  $\text{End}(E_0)$  with  $(P, Q)$  a basis of  $E_0[N]$  and  $\ell$  such that  $\gcd(\ell, N) = 1$  and  $\ell(N - \ell) > p$  with  $N$  smooth.

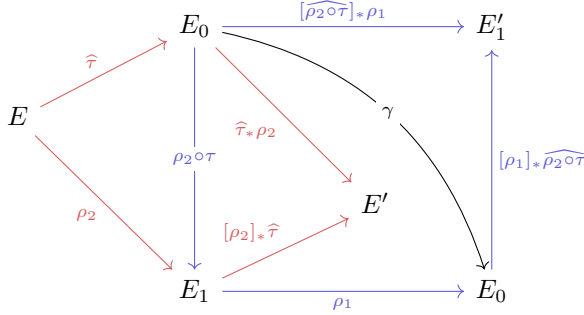
**Output:**  $\tau, \widehat{\rho} : E_0 \rightarrow E$  isogenies of respective degree  $\ell$  and  $N - \ell$ , together with  $I_\tau$  and  $I_{\widehat{\rho}}$  their ideals.

- 1:  $\gamma \leftarrow \mathbf{FullRepresentInteger}(\mathfrak{D}_0, \ell(N - \ell))$
  - 2:  $\mathbf{B} \leftarrow \{([- \ell]P, \gamma(P)), ([- \ell]Q, \gamma(Q))\}$
  - 3:  $F \leftarrow \mathbf{HDKernelToIsogeny}(E_0^2, \mathbf{B})$
  - 4:  $I_\tau \leftarrow \mathcal{O}_0\gamma + \mathcal{O}_0\ell$
  - 5:  $I_{\widehat{\rho}} \leftarrow \mathcal{O}_0\overline{\gamma} + \mathcal{O}_0(N - \ell)$
  - 6: **return**  $\tau, \widehat{\rho}, I_\tau, I_{\widehat{\rho}}$   $\triangleright \tau(-) = F(-, 0)_1$  and  $\widehat{\rho}(-) = -F(0, -)_1$
- 

The concept behind **ExtKaniDoublePath** closely resembles that of **KaniDoublePath**, albeit with a slight variation. Instead of operating with  $\gamma \in \text{End}(E_0)$  of norm  $\ell(N - \ell)$ , **ExtKaniDoublePath** involves working with  $\gamma \in \text{End}(E_0)$  of norm  $\ell(N' - \ell)(N - \ell(N' - \ell))$ , where  $N$  and  $N'$  are smooth. Consequently, we have  $\deg(\rho) = (N' - \ell)(N - \ell(N' - \ell))$ . Both  $\tau$  and  $\widehat{\rho}$  are computed by applying Kani’s Lemma twice:

1. Initially, we decompose  $\gamma$  into  $\gamma = \rho_1 \circ \rho_2 \circ \tau$  where  $\rho_1$  has degree  $N - \ell(N' - \ell)$  and  $\rho_2 \circ \tau$  has degree  $\ell(N' - \ell)$ , and we assess  $\rho_2 \circ \tau$  over  $E_0[N']$ .
2. Subsequently, we further break down  $\rho_2 \circ \tau$  of degree  $\ell(N' - \ell)$  into  $\tau$  and  $\widehat{\rho}_2$  of degree  $\ell$  and  $N' - \ell$  respectively, allowing for the computation of  $\widehat{\rho}$  as a composition of  $\widehat{\rho}_1$  and  $\widehat{\rho}_2$ .

You may find below the commutative diagram of the **ExtKaniDoublePath**. The first use of Kani’s Lemma is in blue and the second is in red.




---

**Algorithm 2. ExtKaniDoublePath**

---

**Input:**  $\mathfrak{D}_0$  an evaluation basis of  $\text{End}(E_0)$  with  $\langle P, Q \rangle$  a basis of  $E_0[N]$ ,  $\langle P', Q' \rangle$  a basis of  $E_0[N']$  and  $\ell$  such that  $\gcd(\ell, N) = \gcd(\ell, N') = 1$  and  $\ell(N' - \ell)(N - \ell(N' - \ell)) > p$  with  $N, N'$  smooth.

**Output:**  $\tau, \widehat{\rho} : E_0 \rightarrow E$  isogenies of respective degree  $\ell$  and  $(N' - \ell)(N - \ell(N' - \ell))$ , together with  $I_\tau$  and  $I_{\widehat{\rho}}$  their ideals.

- 1:  $\gamma \leftarrow \mathbf{FullRepresentInteger}(\mathfrak{D}_0, \ell(N' - \ell)(N - \ell(N' - \ell)))$
  - 2:  $\mathbf{B}_1 \leftarrow \{([- \ell(N' - \ell)]P, \gamma(P)), ([- \ell(N' - \ell)]Q, \gamma(Q))\}$
  - 3:  $F_1 \leftarrow \mathbf{HDKernelToIsogeny}(E_0^2, \mathbf{B}_1) \quad \triangleright \tau \circ \rho_2(-) = F_1(-, 0)_1$
  - 4: Find  $E_1$  the codomain of  $(\widehat{\rho}_1)$
  - 5:  $\mathbf{B}_2 \leftarrow \{([N' - \ell]P', \tau \circ \rho_2(P')), ([N' - \ell]Q', \tau \circ \rho_2(Q'))\}$
  - 6:  $F_2 \leftarrow \mathbf{HDKernelToIsogeny}(E_0 \times E_1, \mathbf{B}_2)$
  - 7:  $I_\tau \leftarrow \mathcal{O}_0 \gamma + \mathcal{O}_0 \ell$
  - 8:  $I_{\widehat{\rho}} \leftarrow \mathcal{O}_0 \widehat{\gamma} + \mathcal{O}_0(N' - \ell)(N - \ell(N' - \ell))$
  - 9: **return**  $\tau, \widehat{\rho}, I_\tau, I_{\widehat{\rho}} \quad \triangleright \tau(-) = -F_2(-, 0)_1$  and  $\widehat{\rho}(-) = F_2(0, -)_1 \circ -F_1(0, -)_1$
- 

*Remark 1.* In **KaniDoublePath** and **ExtKaniDoublePath**, and in other algorithms throughout this paper, we return isogenies and their ideal representations. In practice, during implementation, instead of returning an isogeny, one usually returns its evaluation on some relevant torsion point basis. These torsion point images are used later on to evaluate the isogeny on points lying in the same torsion group.

We will rely on the following assumptions when discussing the security of SQIPrime.

**Assumption 1.** *The distribution of  $E$  the codomain of  $\tau$  and  $\widehat{\rho}$ , returned by **KaniDoublePath**  $(N, P, Q, \ell)$  with  $\ell$  a random prime smaller than  $\sqrt{p}$  is computationally indistinguishable from the distribution of  $E$  sampled randomly among all supersingular curves.*

**Assumption 2.** *The distribution of an isogeny  $\tau : E_0 \rightarrow E$  returned by **ExtKaniDoublePath**  $(N, P, Q, N', P', Q', \ell)$  with  $\ell < \sqrt{p}$  a random prime is computationally indistinguishable from the distribution of  $\tau : E_0 \rightarrow E$  sampled randomly among isogenies of degree  $\ell$  and of domain  $E_0$ .*

### 3.2 KernelToIdeal for Generic Degree Isogenies

Looking at the details of **KernelToIdeal** [11, Algorithm 9], we see that it makes extensive use of discrete logarithms over  $E[d]$ , with  $d$  being the degree of the isogeny for which the representing ideal is being computed. To be efficient via standard methods (i.e. Pohlig-Hellman), this method requires  $d$  being smooth. We therefore need another method for isogenies of generic degree. The idea proposed by Leroux in [33] is to use the knowledge of the endomorphism ring of  $E$  to construct a *precomputed basis* of  $E[d]$ .

**Definition 1.** *Let  $E$  be any supersingular curve. The tuple  $(P, Q, \iota, I_P)$  is a **precomputed basis** of  $E[d]$  if the following conditions are satisfied:*

- $P, Q \in E$  form a basis of  $E[d]$ .
- $\iota \in \text{End}(E)$  and  $\iota(P) = Q$ .
- $I_P$  is the ideal corresponding to the isogeny of kernel  $\langle P \rangle$ .

Knowledge of an evaluation basis  $\mathfrak{D}_E$  of  $\text{End}(E)$  enables us to construct a precomputed basis using the **FindPrecomputedBasis** algorithm (Algorithm 3), proposed in [33]. In our case, we apply it to the curve  $E_0$ , where we can use the (heuristic) **FullRepresentInteger** algorithm to efficiently sample endomorphisms in  $\mathcal{O}_0$  with the desired norm  $dN$  where  $N$  is co-prime to  $d$  and  $p \ll dN$ .

Using a precomputed basis, we can compute ideals from a kernel generator  $K \in E[d]$  by applying the following lemma.

**Lemma 1.** *Let  $(P, Q, \iota, I_P)$  be a precomputed basis of  $E[d]$  and let  $K = [a]P + [b]Q$  be a point in  $E[d]$ . Then the representing ideal of the isogeny  $\phi_K : E \rightarrow E/\langle K \rangle$  is given by  $I_K = [a + b\epsilon(\iota)]_* I_P$  where  $\epsilon : \mathcal{O}_E \leftrightarrow \text{End}(E)$ .*

*Proof.* This comes from the fact that  $\langle K \rangle = \langle [a]P + [b]Q \rangle = \langle [a]P + [b]\iota(P) \rangle = [a + b\iota]\langle P \rangle$ , meaning that  $\phi_K = [a + b\epsilon(\iota)]_* \phi_P$ . We then get the desired result through the Deuring correspondence. □

---

**Algorithm 3. FindPrecomputedBasis**

---

**Input:**  $\mathfrak{D}_E = (\{b_i\}_{i=1}^4, \epsilon)$  an evaluation basis of  $\text{End}(E)$  with  $d$  prime.

**Output:**  $(P, Q, \iota, I_P)$  a precomputed basis of  $E[d]$ .

- 1: Sample a random  $R \in E[d]$
  - 2: Sample  $\alpha \in \mathcal{O}_E$  such that  $\gcd(n(\alpha), d^2) = d$
  - 3: **if**  $\epsilon^{-1}(\alpha)(R) = 0$  **then** go to step 1
  - 4:  $P \leftarrow \epsilon^{-1}(\alpha)(R)$
  - 5:  $I_P \leftarrow \mathcal{O}_E \bar{\alpha} + \mathcal{O}_E d$
  - 6: Sample  $\gamma \in \mathcal{O}_E$  such that  $\gcd(n(\gamma), d) = 1$
  - 7: **if**  $P$  and  $\epsilon^{-1}(\gamma)(P)$  are linearly dependent, **then** go to step 6
  - 8: **return**  $P, \epsilon^{-1}(\gamma)(P), \epsilon^{-1}(\gamma), I_P$
- 

We can thus compute the ideals corresponding to a kernel of generic order. Nevertheless, the method that we presented here requires knowing  $\mathfrak{D}_E$ . Most of the time, the curve  $E$  is obtained by computing an isogeny  $\phi : E_0 \rightarrow E$ . With the knowledge of  $\mathfrak{D}_0$  and  $\phi : E_0 \rightarrow E$ , one can recover  $\mathfrak{D}_E$ , and hence determine a precomputed basis of  $E[d]$  using the **FindPrecomputedBasis** algorithm. Even though this is already efficient, in Corollary 1, we describe a faster and more convenient method to translate a kernel generator  $K \in E[d]$  into an ideal knowing a precomputed basis of  $E_0[d]$ ,  $\phi : E_0 \rightarrow E$  of degree co-prime to  $d$  and its corresponding ideal  $I_\phi$ .

**Corollary 1.** *Let  $(P, Q, \iota, I_P)$  be a precomputed basis of  $E_0[d]$  and let  $\phi : E_0 \rightarrow E$  be an isogeny of degree  $q$  with corresponding ideal  $I_\phi$  such that  $d$  and  $q$  are co-prime. Let  $S, T \in E$  be the respective images of  $P$  and  $Q$  by  $\phi$  and let  $K = [a]S + [b]T$  be a point in  $E[d]$ . Then  $I_K = [(a + b\epsilon(\iota))I_\phi]_* I_P$ .*

*Proof.* Similarly to Lemma 1, we have that

$$\begin{aligned} \langle K \rangle &= [q]\langle K \rangle = \phi \widehat{\phi} \langle [a]S + [b]T \rangle = \phi \langle [a]\widehat{\phi}(S) + [b]\widehat{\phi}(T) \rangle = \phi \langle [aq]P + [bq]Q \rangle \\ &= \phi \langle [a]P + [b]Q \rangle = \phi \langle [a]P + [b]\iota(P) \rangle = \phi \circ [a + b\iota] \langle P \rangle, \end{aligned}$$

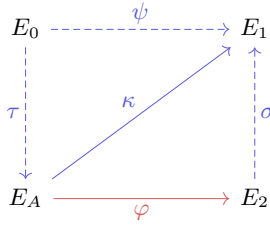
i.e.  $\phi_K = [\phi \circ (a + b\iota)]_* \phi_P$  and thus  $I_K = [(a + b\epsilon(\iota))I_\phi]_* I_P$ . □

It's worth noting that [33] proposes using  $\phi$  to directly generate a precomputed basis over  $E$ . Specifically, if  $(P, Q, \iota, I_P)$  represents a precomputed basis over  $E_0[d]$ , then  $(\phi(P), [\text{deg}(\phi)]\phi(Q), \theta, [I_\phi]_* I_P)$  constitutes a precomputed basis of  $E[d]$  with  $\theta = \phi \circ \iota \circ \widehat{\phi}$ . The significant advantage of Corollary 1 lies in its exclusive use of endomorphisms over  $E_0$  rather than over  $E$ . This characteristic aligns more closely with our requirements in SQIPrime, making it better suited for our purposes.

## 4 SQIPrime4D: SQIPrime in Dimension 4

As previously stated in the introduction, SQIPrime4D further expands the use of Kani's Lemma to both KeyGen and Commit. Moreover, the challenge isogeny has

non-smooth degree. Only the kernel of the challenge isogeny is sampled by the verifier. The challenge isogeny  $\varphi : E_A \rightarrow E_2$  is computed by the prover, who then appends the usual response isogeny  $\sigma : E_2 \rightarrow E_1$  to it to get  $\kappa := \sigma \circ \varphi : E_A \rightarrow E_1$ . The high dimensional representation of  $\kappa$  is returned to the verifier. Figure 2 illustrates the architecture of SQIPrime4D.



**Fig. 2.** Diagram of SQIPrime4D, prover in blue and verifier in red. Dashed isogenies are not shared. (Color figure online)

The public parameters of SQIPrime4D are defined as:

- $p$  a prime number of the form  $p = 2^\alpha f - 1 \simeq 2^{2\lambda}$  and such that  $p = 2Nq + 1$ , with  $q \simeq 2^\lambda$ . We discuss in Sect. 8 how to efficiently compute such primes.
- $P_0, Q_0$  a basis of  $E_0[2^\alpha]$ .
- $(P, Q, \iota, I_{[N]P})$  which is almost a precomputed basis over  $E_0[Nq]$ . (It is if we use  $I_P$  instead of  $I_{[N]P}$  but this ideal is more adapted to SQIPrime4D.)
- $\beta$  an integer of the form  $\beta = 2\lambda + c \log(\lambda)$  with  $c$  a small constant. (See Sect. 4.2 for more details.)

They are constructed using the Setup algorithm described in Algorithm 4.

---

**Algorithm 4. SQIPrime4D.Setup**

---

**Input:**  $1^\lambda$ .

**Output:**  $\text{pp} = (p, \alpha, q, N, (P_0, Q_0), (P, Q, \iota, I_{[N]P}), \beta)$ .

- 1: Take  $p$  a prime of the form  $p = 2^\alpha f - 1 \simeq 2^{2\lambda}$  such that  $p - 1 = 2Nq$  with  $q \simeq 2^\lambda$  prime and  $N$  co-prime to  $q$
  - 2:  $P_0, Q_0 \leftarrow \text{CanonicalTorsionBasis}(E_0, 2^\alpha)$
  - 3:  $(P, Q, \iota, I_P) \leftarrow \text{FindprecomputedBasis}(\mathfrak{O}_0, qN)$
  - 4: Compute  $I_{[N]P} = I_P + \mathcal{O}_0q$
  - 5:  $\beta \leftarrow \lceil 2\lambda + c \log_2(\lambda) \rceil$
  - 6:  $\text{pp} \leftarrow (p, (P_0, Q_0), (P, Q, \iota, I_{[N]P}), \beta)$
  - 7: **return**  $\text{pp}$
- 

At a high level, the subroutines of SQIPrime4D are as follows.

- **KeyGen**: Compute  $\tau : E_0 \rightarrow E_A$  together with its corresponding ideal  $I_\tau$  using **KaniDoublePath**. Additionally, compute a matrix  $\mathbf{M}$  and use it to mask the image through  $\tau$  of a precomputed basis of degree  $qN$ , with  $q \simeq 2^\lambda$ . The curve  $E_A$  and the masked basis form the public key, while  $\tau, I_\tau$  and the matrix  $\mathbf{M}$  form the secret key.
- **Commit**: The prover computes an isogeny  $\psi : E_0 \rightarrow E_1$  with **KaniDoublePath** together with its ideal  $I_\psi$  and shares  $E_1$ .
- **Challenge**: The verifier samples a random scalar  $a \in \mathbb{Z}_q$  and returns it to the prover. This scalar defines a point  $C_a = P + [a]Q$  where  $P, Q$  is a specified basis of  $E_A[q]$ .
- **Response**: Using the precomputed basis over  $E_0$  and its knowledge of  $I_\tau$ , the prover retrieves  $I_\varphi$ , the ideal corresponding to the challenge isogeny  $\varphi : E_A \rightarrow E_2$  whose kernel is given by  $\ker(\varphi) = \langle C_a \rangle$ . Using **RandomEquivalentIdeal**, they compute a short  $(\mathcal{O}_2, \mathcal{O}_1)$ -ideal  $I_\sigma$  corresponding to an isogeny  $\sigma : E_2 \rightarrow E_1$ , and construct  $\kappa = \sigma \circ \varphi$ , evaluate it using **EvalTorsion** and send this evaluation of  $\kappa$  as the response to the verifier.
- **Verify**: The verifier receives  $\kappa$  and checks using Kani’s Lemma that it is valid by verifying that it is an isogeny from  $E_A$  to  $E_1$  and that  $\kappa(C_a) = 0$ .

#### 4.1 Key Generation and Commitment

Both key generation and commitment consist essentially in using **KaniDoublePath**. We take a random prime  $\ell$  smaller than  $\sqrt{p}$  and use the **KaniDoublePath** with an endomorphism of norm  $\ell(2^\alpha - \ell)$  to retrieve  $\tau$  in the case of **SQIPrime.KeyGen** (Algorithm 5) and  $\psi$  in **SQIPrime.Commit**. (Algorithm 6). The only significant differences between the key and commitment generation is that during the key generation, we additionally compute a masked basis of  $E_A[Nq]$ . To do so, we compute the image of  $(P, Q)$  through the isogeny  $\tau$  and use a random matrix  $\mathbf{M} \in \text{GL}_2(Nq)$  to mask the torsion points. Note that this masking makes of  $R, S$  a random basis of  $E_A[Nq]$ .

---

#### Algorithm 5. SQIPrime4D.KeyGen

---

**Input:**  $\text{pp} = (p, \alpha, q, N, (P_0, Q_0), (P, Q, \iota, I_{[N]P}), \beta)$ .

**Output:**  $\text{sk} = (\tau, I_\tau, \mathbf{M}), \text{pk} = (E_A, (R, S))$ .

- 1: Sample  $\ell_A \neq 2$  a random prime smaller than  $\sqrt{p}$  such that  $\ell_A$  co-prime with  $q$
  - 2:  $\tau, *, I_\tau, * \leftarrow \mathbf{KaniDoublePath}(2^\alpha, P_0, Q_0, \ell_A)$
  - 3: Compute  $E_A = \text{Im}(\tau)$
  - 4: Sample a random matrix  $\mathbf{M} \in \text{GL}_2(Nq)$
  - 5:  $\begin{pmatrix} R \\ S \end{pmatrix} \leftarrow \mathbf{M} \begin{pmatrix} \tau(P) \\ \tau(Q) \end{pmatrix}$
  - 6: **return**  $(\tau, I_\tau, \mathbf{M}), (E_A, (R, S))$
-



---

**Algorithm 6. SQIPrime4D.Commit**

---

**Input:**  $\text{pp} = (p, \alpha, q, N, (P_0, Q_0), (P, Q, \iota, I_{[N]P}), \beta)$ .

**Output:**  $\text{sec} = (\psi, I_\psi)$ ,  $\text{com} = E_1$ .

- 1: Take  $\ell_1 \neq 2$  a random prime smaller than  $\sqrt{p}$  such that  $\ell_1$  co-prime with  $q$
  - 2:  $\psi, *, I_\psi, * \leftarrow \mathbf{KaniDoublePath}(2^\alpha, P_0, Q_0, \ell_1)$
  - 3: Compute  $E_1 = \text{Im}(\psi)$
  - 4: **return**  $(\psi, I_\psi), (E_1)$
- 

## 4.2 Challenge and Response

**Challenge.** As touched on earlier, our challenge is significantly different from the challenge of SQISign and SQISignHD, as the evaluation of the challenge isogeny has been moved from the verifier to the prover. This adjustment is necessary since the verifier lacks an efficient means to evaluate this isogeny, as it only has access to the kernel representation of  $\varphi$ , whose degree is not smooth. The prover uses the ideal representation to construct a high dimension representation of  $\varphi$  that is then sent to the verifier together with the high dimension representation of the answer isogeny  $\sigma$ . Thus, instead of providing an isogeny of smooth degree, the challenger simply sends a challenge point  $C_a \in E_A[q]$ . This point is given as  $a \in \mathbb{Z}_q$  such that  $C_a = [N](R + [a]S)$  where  $R, S$  is the basis of  $E_A[Nq]$  included in the public key. This point is the generator of the kernel of  $\varphi : E_A \rightarrow E/\langle C_a \rangle = E_2$ . We have  $q \simeq 2^\lambda$  possible challenge isogenies.

**Response.** In line with SQISignHD, our objective is to compute an isogeny  $\sigma : E_2 \rightarrow E_1$ . However, the verifier lacks knowledge of  $E_2$ . An initial idea might be to provide the verifier with an HD representation of  $\varphi$ , allowing him to check that the kernels match. However, this approach requires knowledge of a map between  $E_0$  and  $E_2$  (or  $E_A$  and  $E_2$ ), which is challenging to construct.<sup>3</sup> Instead of sending  $\sigma$  and  $\varphi$  separately, the idea is to send  $\kappa = \sigma \circ \varphi$  and use Kani’s Lemma over  $\kappa$  to prove that  $\kappa$  factors through  $\varphi$ , utilising the fact that  $\ker(\kappa) \cap E_A[q] = \ker(\varphi)$ .

First, one adapts Corollary 1 to compute  $I_{C_a} = I_\varphi$ . Upon receiving the challenge  $\text{Chal} = a$ , the prover finds  $b, c \in \mathbb{Z}_q$  such that  $C_a = [N]([b]\tau(P) + [c]\tau(Q))$ . These scalars are given by  $\begin{pmatrix} b \\ c \end{pmatrix} = \mathbf{M}^\top \begin{pmatrix} 1 \\ a \end{pmatrix}$ . One then recovers  $I_{C_a}$  as

$$I_{C_a} = [(b + c\epsilon(\iota))I_\tau]_* I_{[N]P}$$

One then computes the  $(\mathcal{O}_2, \mathcal{O}_1)$ -ideal  $\overline{I_{C_a} I_\tau} I_\varphi$  and finds an equivalent short  $(\mathcal{O}_2, \mathcal{O}_1)$ -ideal  $J$  using **RandomEquivalentIdeal**. The ideal  $J$  corresponds to an isogeny  $\sigma : E_2 \rightarrow E_1$  of degree  $d$  as shown in Fig. 2, with  $d$  such that  $2^\beta - qd$  can be written as the sum of two squares. One sufficient condition

---

<sup>3</sup> We could use the KLPT algorithm followed by the **IdealToKernel** algorithm, but avoiding this algorithm was a primary motivation behind the development of SQISignHD.

is to ask for  $2^\beta - qd = 1 \pmod 4$  and to be prime. Following the discussion in [11, Section 4.2] and by using the sampling method proposed in [11, Section E.2], we expect to find a valid  $J$  after sampling  $O(\lambda)$  times. Moreover, we require that  $d$  is co-prime to  $q$ . This is to prevent backtracking when composing  $\sigma$  and  $\varphi$ . Since  $q$  has very few prime factors in our case, then a few supplementary samples will allow to ensure that  $d$  and  $q$  are co-prime. For the suggested parameters (Sect. 8), the worst case is when  $\lambda = 192$  where  $q = 3 \cdot 7 \cdot 4803463386334137403 \cdot 116682096886878909945888202135243873061$  and that the probability that a random number shares a prime factor with  $q$  is at most 0.47. Note that  $\beta$  can be as large as  $2\alpha \approx 2 \log p$ , which means there is more than enough room to sample  $J$  with the requirements above. In practice,  $\beta = 2\lambda + c \log(\lambda) \ll 2^{3\lambda}$  where  $c$  is a small constant is sufficient.

The final response is composed of the evaluation of the isogeny  $\kappa = \sigma \circ \varphi$  on  $E_A[2^\alpha]$  and on the point  $C_2 = [a]R - S$ , together with the degree  $d$  of  $\sigma$ . To do so, one generates a basis of  $E_A[2^\alpha]$  using **CanonicalTorsionBasis**, one uses **EvalTorsion** to evaluate  $\kappa$  on the generated basis and  $C_2$ . The point  $\kappa(C_2)$  is used to ensure the soundness of our verification. It is important to note that  $C_2$  satisfies  $\langle C_a, [N]C_2 \rangle = E_A[q]$ .

---

**Algorithm 7. SQIPrime4D.Response**

---

**Input:**  $\text{pp} = (p, \alpha, q, N, (P_0, Q_0), (P, Q, \iota, I_{[N]P}), \beta)$ ,  $\text{sk} = (\tau, I_\tau, \mathbf{M})$ ,  $\text{pk} = (E_A, (R, S))$ ,  $\text{sec} = (\psi, I_\psi)$ ,  $\text{com} = E_1$ ,  $\text{chal} = a$ .

**Output:**  $\text{res} = (T, U, V, d)$  with  $T, U \in E_1[2^\alpha]$ ,  $V \in E_1[Nq]$  and  $d$  the degree of  $\sigma$ .

- 1:  $\begin{pmatrix} b \\ c \end{pmatrix} \leftarrow \mathbf{M}^\top \begin{pmatrix} 1 \\ a \end{pmatrix}$
  - 2:  $I_{C_a} \leftarrow [(b + ct)I_\tau]_* I_{[N]P}$
  - 3:  $J \leftarrow \mathbf{RandomEquivalentIdeal}(\overline{I_{C_a} I_\tau} I_\psi) \quad d \leftarrow n(J)$
  - 4: **if**  $\gcd(d, q) \neq 1$  **or**  $2^\beta - dq \neq 1 \pmod 4$  **or**  $2^\beta - dq$  is composite, **go back to Step 3**
  - 5:  $X, Y \leftarrow \mathbf{CanonicalTorsionBasis}(E_A, 2^\alpha)$
  - 6:  $C_2 \leftarrow [a]R - S$
  - 7:  $T, U, V \leftarrow \mathbf{EvalTorsion}(\mathcal{D}_0, \tau, I_\tau, \psi, I_\psi, I_{C_a} J, qd, \{X, Y, C_2\})$
  - 8: **return**  $\text{res} = (T, U, V, d) \quad \triangleright T = \kappa(X), U = \kappa(Y), V = \kappa(C_2)$
- 

**4.3 Verification**

Upon receiving  $T, U, V, d$ , we want to verify that the following statement holds: *the torsion points we received define a high dimensional representation of an isogeny  $\kappa : E_A \rightarrow E_1$  of degree  $dq$  such that  $d$  and  $q$  are co-prime and the isogeny  $\kappa$  factors through  $\varphi$ , meaning that  $\ker(\kappa)[q] = \langle C_a \rangle$ .*

To perform this verification efficiently, we use Kani’s Lemma to construct the isogeny  $F : E_1^2 \times E_A^2 \rightarrow E_A^2 \times E_1^2$  given by the following diagram and matrices:

$$\begin{array}{ccc}
 E_A^2 & \xrightarrow{\Sigma} & E_1^2 \\
 \eta \downarrow & & \downarrow \eta \\
 E_A^2 & \xrightarrow{\Sigma} & E_1^2
 \end{array}
 \quad
 F := \begin{pmatrix} \tilde{\Sigma} & -\tilde{\eta} \\ \eta & \Sigma \end{pmatrix} = \begin{pmatrix} \hat{\kappa} & 0 & -a_1 & -a_2 \\ 0 & \hat{\kappa} & a_2 & -a_1 \\ a_1 & -a_2 & \kappa & 0 \\ a_2 & a_1 & 0 & \kappa \end{pmatrix}$$

where  $\eta := \begin{pmatrix} a_1 & -a_2 \\ a_2 & a_1 \end{pmatrix}$  such that  $\deg(\eta) = a_1^2 + a_2^2$ ;  $\Sigma := \text{diag}(\kappa, \kappa)$ . If the parameters allow us to always have enough torsion, that is we always have  $dq < 2^\alpha$  or equivalently  $\beta = \alpha$ , then  $F$  can be computed on one go and its kernel is given by  $\ker(F) = \{(\Sigma(P), -\eta(P)) \mid P \in E_A^2[2^\beta]\}$ . If the parameters do not allow this, then we split the isogeny  $F : E_1^2 \times E_A^2 \rightarrow E_A^2 \times E_1^2$  into two isogenies  $F_1 : E_1^2 \times E_A^2 \rightarrow \Delta$  and  $F_2 : \Delta \rightarrow E_A^2 \times E_1^2$  where  $\Delta$  is an abelian surface,  $F = F_2 \circ F_1$  with  $\deg(F_i) = 2^{\beta_i}$  ( $\beta_1 + \beta_2 = \beta$ ),  $\ker(F_1) = \{(\Sigma(P), -\eta(P)) \mid P \in E_A^2[2^{\beta_1}]\}$  and  $\ker(\widetilde{F}_2) = \{(\Sigma(P), \tilde{\eta}(P)) \mid P \in E_A^2[2^{\beta_2}]\}$ , similarly to SQISignHD<sup>4</sup>. We then use the following property: let  $X \in E_A$  be a point of odd order, then

$$F \begin{pmatrix} 0 \\ 0 \\ X \\ 0 \end{pmatrix} = \begin{pmatrix} [-a_1]X \\ [a_2]X \\ Y \\ 0 \end{pmatrix} \iff [2^{\beta_2}]F_1 \begin{pmatrix} 0 \\ 0 \\ X \\ 0 \end{pmatrix} = \widetilde{F}_2 \begin{pmatrix} [-a_1]X \\ [a_2]X \\ Y \\ 0 \end{pmatrix}.$$

We use this equivalence on the points  $C_a$  and  $C_2$  of respective order  $q$  and  $Nq$ .

**Proposition 1.** *Let pp, pk, com, chal be a valid public key, commitment, and challenge of SQIPrime4D and let  $P, Q$  be the canonical basis of  $E_A[2^\alpha]$ . Let  $\overline{\text{Res}}$  be a potential response. **SQIPrime4D.Verify**(pp, pk, com, chal,  $\overline{\text{Res}}$ ) = 1 implies that  $\overline{\text{Res}} = (\overline{T}, \overline{U}, \overline{V}, \overline{d})$  is such that:*

- $(P, Q, \overline{T}, \overline{U})$  is a high dimension representation of an isogeny  $\kappa : E_A \rightarrow E_1$  of degree  $q\overline{d}$ .
- $\ker(\kappa) \cap E_A[q] = \langle C_a \rangle$ .

*Proof.* Our proof takes inspiration from [11, Section E.5]. In fact if we assume that **SQIPrime.Verify**(pp, pk, pub, chal,  $\overline{\text{Res}}$ ) = 1, then  $\overline{T}, \overline{U}, \overline{V}$  are in  $E_1$ ,  $\overline{F}_1$  and  $\overline{F}_2$  are well-defined and have the same codomain, and the following holds:

$$\begin{aligned}
 [2^{\beta_2}]\overline{F}_1(0, 0, C_a, 0) &= \widetilde{F}_2([-a_1]C_a, [a_2]C_2, 0, 0) \implies \overline{F}(0, 0, C_a, 0) = ([-a_1]C_a, [a_2]C_2, 0, 0) \\
 [2^{\beta_2}]\overline{F}_1(0, 0, C_2, 0) &= \widetilde{F}_2([-a_1]C_a, [a_2]C_2, \overline{V}, 0) \implies \overline{F}(0, 0, C_2, 0) = ([-a_1]C_a, [a_2]C_2, \overline{V}, 0).
 \end{aligned}$$

From the isogeny  $\overline{F}$ , using  $\iota_i$  and  $\rho_j$  the standard injections/restrictions of product spaces, we can construct 16 elliptic curve isogenies  $\overline{F}_{i,j} = \rho_i \circ \overline{F} \circ \iota_j$  with  $1 \leq i, j \leq 4$  such that for all  $j = 1, \dots, 4$ :

<sup>4</sup> A slight change in the prime used in SQISignHD was suggested in [24] in order to avoid splitting the high dimensional isogeny, in the hope for a better efficiency, but we are not aware of any implementation of this variant.

**Algorithm 8. SQIPrime4D.Verify**

**Input:**  $\text{pp} = (p, (P_0, Q_0), (P, Q, \iota, I_{[N]P}), \beta)$ ,  $\text{pk} = (E_A, R, S)$ ,  $\text{com} = E_1$ ,  $\text{chal} = a$ ,  $\text{res} = (T, U, V, d)$ .

**Output:** 0 or 1.

- 1: **if** one of the points  $T, U, V$  is not in  $E_1$  **or**  $\gcd(d, q) \neq 1$ , **return** 0
- 2:  $\beta_1 \leftarrow \lfloor \frac{\beta}{2} \rfloor$ ,  $\beta_2 \leftarrow \lceil \frac{\beta}{2} \rceil$ ,  $k_1 \leftarrow 2^{\alpha - \beta_1}$ ,  $k_2 \leftarrow 2^{\alpha - \beta_2}$
- 3:  $(a_1, a_2) \leftarrow \text{Cornacchia}(2^\beta - qd)$
- 4: Compute  $\eta$  and  $\tilde{\eta}$
- 5: Compute  $\{P_i\}_{0 \leq i \leq 4}$  a basis of  $E_A^2[2^\alpha]$  ▷ Using **CanonicalTorsionBasis**
- 6:  $B_1 \leftarrow \{([k_1]\Sigma(P_i), [-k_1]\eta(P_i))\}_{0 \leq i \leq 4}$  ▷  $\Sigma(P_i)$  computed using  $T, U$
- 7:  $B_2 \leftarrow \{([k_2]\Sigma(P_i), [k_2]\tilde{\eta}(P_i))\}_{0 \leq i \leq 4}$
- 8:  $F_1 \leftarrow \text{HDKernelToIsogeny}(B_1)$
- 9:  $\widetilde{F}_2 \leftarrow \text{HDKernelToIsogeny}(B_2)$
- 10: **if**  $\text{codomain}(F_1) \neq \text{codomain}(\widetilde{F}_2)$  **do return** 0 ▷ Do as [11, Section F.3]
- 11:  $C_a \leftarrow [N](R + [a]S)$ ,  $C_2 \leftarrow ([a]R - S)$
- 12:  $b_1 \leftarrow [2^{\beta_2}]F_1(0, 0, C_a, 0) \stackrel{?}{=} \widetilde{F}_2([-a_1]C_a, [a_2]C_a, 0, 0)$
- 13:  $b_2 \leftarrow [2^{\beta_2}]F_1(0, 0, C_2, 0) \stackrel{?}{=} \widetilde{F}_2([-a_1]C_2, [a_2]C_2, V, 0)$
- 14: **return**  $b_1 \wedge b_2$

$$\sum_{i=1}^4 \deg(\overline{F}_{i,j}) = \deg(\overline{F}) = 2^\beta$$

We focus on the case when  $j = 3$ . We want to demonstrate that for  $i = 1, 2$ , and 4,  $F_{i,3} = [b_i]$ , with  $b_i$  being  $-a_1$ ,  $a_2$ , and 0, respectively. To achieve this, we utilize the Cauchy interpolation theorem. By applying the triangular inequality, we have:

$$\text{for } i = 1, 2, 4, \deg(\overline{F}_{i,3} - [b_i]) \leq 4 \cdot 2^\beta \approx 2^{2\lambda + c \log(\lambda) + 2} \ll 2^{3\lambda}.$$

We know that  $\overline{F}_{i,3} = [b_i]$  for all points generated by  $\langle C_a, C_2 \rangle$ , i.e., for  $Nq^2 \approx 2^{3\lambda}$  points. Thus,  $\overline{F}_{1,3} = [a_1]$ ,  $\overline{F}_{2,3} = [-a_2]$ , and  $\overline{F}_{4,3} = 0$ . Since  $\overline{F}(0, 0, C_a, 0) = ([-a_1]C_a, [a_2]C_2, 0, 0)$ , we deduce that  $\overline{F}_{3,3}$  is an isogeny of degree  $q\overline{d}$  between  $E_A$  and  $E_1$  such that  $\overline{F}_{3,3}(C_a) = 0$ . Since  $\overline{d}$  and  $q$  are co-prime, then  $\ker(\overline{F}_{3,3}) \cap E_A[q] = \langle C_a \rangle$ .  $\square$

## 5 Security Analysis of SQIPrime4D

We now prove that the SQIPrime4D identification protocol described in the Sect. 4 is a  $\Sigma$ -protocol. To do so, we have to show that SQIPrime4D has special soundness and is Honest Verifier Zero Knowledge (HVZK). Once both are proven, applying the Fiat-Shamir transform [23] over SQIPrime4D will result in a digital signature scheme that is UU-CMA in the ROM. The extractor is constructed as follows.

**Proposition 2.** *Let  $(E_1, \text{chal}_1, T_1, U_1, V_1, d_1)$  and  $(E_1, \text{chal}_2, T_2, U_2, V_2, d_2)$  be 2 transcripts with identical commitment  $E_1$  and  $\text{chal}_1 \neq \text{chal}_2$ . There exists an extractor  $\mathcal{E}$  that, given both transcripts, can efficiently solve the one endomorphism problem (Problem 1) over  $E_A$ , i.e. find  $\theta_A \in \text{End}(E_A)$  a non-trivial endomorphism.*

*Proof.* Our proof is very similar to [11, Proposition 17]. We can use  $T_1, U_1$  to compute a high dimension representation of  $\kappa_1 = \sigma_1 \circ \varphi_1$  and  $T_2, U_2$  to compute a high dimension representation of  $\widehat{\kappa_2} = \widehat{\sigma_2 \circ \varphi_2}$ . Then,  $\theta_A = \widehat{\kappa_2} \circ \kappa_1 \in \text{End}(E_A)$  is non-scalar. In fact, let us assume for a moment that  $\theta_A$  is a scalar. Since  $\ker(\kappa_1) \cap E_A[q] = \langle C_{\text{chal}_1} \rangle$  is cyclic,  $\ker(\kappa_2) \cap E_A[q] = \langle C_{\text{chal}_2} \rangle$  is cyclic,  $d_1$  and  $d_2$  are co-prime to  $q$ , then  $\ker(\kappa_1) \cap E_A[q] = \ker(\kappa_2) \cap E_A[q]$ , which implies that  $\langle C_{\text{chal}_1} \rangle = \langle C_{\text{chal}_2} \rangle$ . Hence  $\text{chal}_1 = \text{chal}_2$ , which is a contradiction.  $\square$

The extractor ensures us that SQIPrime4D has special soundness. Similarly to [11, Section 5.2], we construct the simulator under the assumption that we have access to the following oracle.

**Definition 2.** *The Random Uniformly Constrained Good Degree Isogeny Oracle (RUCGDIO) is an oracle that takes as input a supersingular curve  $E$  together with  $P \in E[q]$  and that returns an efficient representation of  $\kappa : E \rightarrow E'$  of degree  $qd$  with  $d$  co-prime with  $q$  and such that:*

- $E'$  is uniformly distributed over all supersingular curves.
- $\kappa$  is uniformly distributed among all isogenies between  $E$  and  $E'$  such that  $P \in \ker(\kappa)$  and such that  $2^3 - qd$  is a prime congruent to 1 modulo 4 with  $d$  co-prime to  $q$ .

**Proposition 3.** *Given  $\text{pp}, \text{pk}$  and  $\text{chal}$ , there exists a simulator  $\mathcal{S}$  with access to a RUCGDIO that simulates transcripts with a distribution that is computationally indistinguishable from the distribution of transcripts of SQIPrime4D, conditioned to  $\text{chal}$ .*

*Proof.* Given  $a \in \mathbb{Z}_q$ , we compute  $C_a = [N](R + [a]S)$ . Calling RUCGDIO over  $E_A$  and  $C_a$ , we retrieve an efficient representation of  $\kappa : E_A \rightarrow E_1$  and use this representation to compute the points  $A = \kappa(X), B = \kappa(Y)$ , and  $Z = \kappa([b]R - [a]S)$  with  $X, Y$  the canonical basis over  $E_A[2^\alpha]$ .

We then simply return the following transcript  $(E_1, a, A, B, Z, \deg(\kappa)/q)$ .

This transcript is computationally indistinguishable from a genuine transcript, as:

- Following Assumption 1, we have that a genuine  $E_1$  or one given by RUDGIO are computationally indistinguishable.
- Following [32, Lemma 3.2.4], a genuine  $\kappa$  or one given by RUDGIO are computationally indistinguishable, and so does  $A, B, Z, \deg(\kappa)/q$ .

$\square$

We now make the following assumption.

**Assumption 3.** *The one endomorphism problem (Problem 1) remains hard even when given access to RUCGDIO.*

Indeed, by definition, RUCGDIO, when given an input  $P$ , generates a random isogeny that factors  $\phi_P$  and that is of good degree. If  $P$  is of smooth order, then RUCGDIO is in fact equivalent to the RUGDIO oracle [11, Definition 5.2.1]. Thus, the arguments of [11, Section 5.3] also applies to RUCGDIO. It is therefore reasonable to assume that RUCGDIO does not help to break the one endomorphism problem.

## 6 SQIPrime2D: SQIPrime in Dimension 2

In this section, we describe a version of SQIPrime which uses only dimension 2 isogenies. As touched on earlier, moving from dimension 4 isogenies to dimension 2 isogenies allows to obtain a more efficient scheme. This time, SQIPrime2D is expected to be more efficient compared to SQISignHD.

### 6.1 High Level Description

Recall the diagram for SQIPrime4D in Fig. 2. In order to represent  $\kappa = \sigma \circ \varphi$  using Kani's Lemma in dimension 2, we need to compute and evaluate an auxiliary isogeny  $\delta : E_A \rightarrow E_\delta$  of degree  $2^\alpha - dq$ . Since the prover knows the endomorphism ring of  $E_A$ , they could in fact compute such an isogeny by using the KLPT algorithm, but this is not an admissible way as we want to avoid using the costly KLPT algorithm.

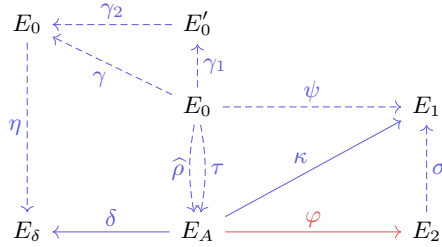
Instead, we will use Kani's Lemma, **KaniDoublePath** and **ExtKaniDoublePath**, together with several other techniques to generate the auxiliary isogeny of degree  $2^\alpha - dq$ . To achieve this goal, we will operate the following change to SQIPrime4D:

*the secret isogeny  $\tau$  will now be of fixed<sup>5</sup> degree  $q$ , which is also the degree of the challenge isogeny  $\varphi$ .*

With that change in mind, we now sketch how one generates an auxiliary isogeny  $\delta : E_A \rightarrow E_\delta$  of degree  $2^\alpha - dq$ . Firstly, one samples an endomorphism  $\gamma \in \text{End}(E_0)$  of degree  $d(2^\alpha - dq)$ , and one evaluates it on the  $2^\alpha$ -torsion. Next, one evaluates  $\tau \circ \hat{\gamma}$  on the  $2^\alpha$ -torsion basis  $\{P_0, Q_0\}$  of  $E_0$ . Write  $\gamma = \gamma_2 \circ \gamma_1$  where  $\gamma_1$  and  $\gamma_2$  have degree  $d$  and  $2^\alpha - dq$  respectively, and let  $E'_0$  be the codomain of  $\gamma_1$ . Let  $\delta : E_A \rightarrow E_\delta$  be the pushforward of  $\gamma_2$  through  $\tau \circ \hat{\gamma}_1$ . Then  $E_0, E'_0, E_A$  and  $E_\delta$  are the vertices of an SIDH square where the degrees are  $dq$  and  $2^\alpha - dq$ . One can hence apply Kani's Lemma to compute the isogeny  $\delta : E_A \rightarrow E_\delta$  and evaluate it on the  $2^\alpha$ -torsion points. This is illustrated in Fig. 3.

For SQIPrime2D, the public parameters are defined as follows:

<sup>5</sup> This already implies that the key recovery problem in SQIPrime4D and SQIPrime2D are different, since the degree of the secret isogeny in SQIPrime4D is random and is not public.



**Fig. 3.** Diagram of SQIPrime2D, prover in blue and verifier in red. Dashed isogenies are not shared. (Color figure online)

- The base prime  $p$  is of the form  $p = 2^\alpha f - 1 = 2Nq + 1 \simeq 2^{2\lambda}$ , with  $q \simeq 2^\lambda$  prime, such that:  $\alpha \geq \lceil \frac{\log_2(p)}{2} + \log_2(q) \rceil + 1$ .
- $P_0, Q_0$  is a basis of  $E_0[2^\alpha]$ .
- $(P, Q, \iota, I_P)$  is a precomputed basis of  $E_0[q]$ .

The computation of the commitment isogeny in SQIPrime2D is identical to that of the secret isogeny in SQIPrime4D, but the key generation, the response and the verification algorithms are modified.

### 6.2 SQIPrime2D Key Generation Algorithm

For the computation of the secret isogeny  $\tau$ , whose degree is  $q$  and is public, we use **ExtKaniDoublePath**. In SQIPrime2D, the points  $R$  and  $S$  are no longer the masked images of  $P$  and  $Q$  by  $\tau$  (as in SQIPrime4D). Instead, they are the masked images by  $\widehat{\rho}$  of the points  $P$  and  $Q$ , where  $\widehat{\rho}$  is the second isogeny computed using **ExtKaniDoublePath**. This change is necessary since  $\deg(\tau) = q$ , which is also the order of the points  $P$  and  $Q$ . We thus have that  $\begin{pmatrix} R \\ S \end{pmatrix} = \mathbf{M}\widehat{\rho}\begin{pmatrix} P \\ Q \end{pmatrix}$ . This time, one also includes  $I_{\widehat{\rho}}$  in the secret key since it is needed when translating the kernel of the non-smooth challenge isogeny into an ideal.

*Remark 2.* With respect to the current state of the art [8, 15, 34, 44] when it comes to the supersingular isogeny problem with torsion point information, there is no known algorithm that exploits the images of torsion points of non-smooth order to weaken the supersingular isogeny problem. All known attacks require the torsion point images to have smooth order. This means that the masking matrix  $\mathbf{M}$  is not really necessary since  $q$  is prime. We nevertheless keep it in order to avoid having to explicitly assume that revealing the non-smooth order torsion point images in clear does not affect the security of the protocol.

### 6.3 SQIPrime2D Response Algorithm

Upon receiving  $\text{Chal} = a \in \mathbb{Z}_q$  from the verifier, the prover computes  $C_a = R + [a]S = [b]\widehat{\rho}(P) + [c]\widehat{\rho}(Q)$ . The prover then calculates  $I_{C_a}$  defined as  $I_{C_a} = [(b + c\epsilon(\iota))I_{\widehat{\rho}}]_* I_P$ .

---

**Algorithm 9. SQIPrime2D.KeyGen**

---

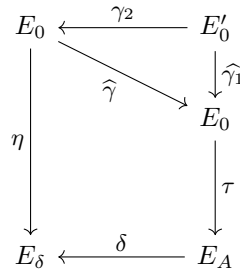
**Input:**  $\text{pp} = (p, \alpha, q, N, (P_0, Q_0), (P, Q, \iota, I_P))$ .

**Output:**  $\text{sk} = (\tau, \hat{\rho}, I_\tau, I_{\hat{\rho}})$ ,  $\text{pk} = (E_A, (R, S))$ .

- 1:  $\tau, \hat{\rho}, I_\tau, I_{\hat{\rho}} \leftarrow \text{ExtKaniDoublePath}(2^\alpha, P_0, Q_0, q)$
  - 2: Compute  $E_A = \text{Im}(\tau)$
  - 3: Sample a random matrix  $\mathbf{M} \in \text{GL}_2(q)$
  - 4:  $\begin{pmatrix} R \\ S \end{pmatrix} \leftarrow \mathbf{M}\hat{\rho}\begin{pmatrix} P \\ Q \end{pmatrix}$
  - 5: **return**  $(\tau, \hat{\rho}, I_\tau, I_{\hat{\rho}}, \mathbf{M}), (E_A, (R, S))$
- 

Next, the prover computes the  $(\mathcal{O}_2, \mathcal{O}_1)$ -ideal  $\overline{I_{C_a} I_\tau} I_\psi$  and locates another small  $(\mathcal{O}_2, \mathcal{O}_1)$ -ideal  $J$  using the **RandomEquivalentIdeal** algorithm. Following [11, Lemma 12], we are assured of the existence of such an ideal with a norm smaller than  $\sqrt{p}$ . Additionally, we require that  $n(J)$  is odd. Notably, this condition is considerably less restrictive than that of SQIPrime4D, as approximately half of all potential isogenies remain valid, compared to only  $1/\log(p)$  in the case of SQIPrime4D. Therefore, we have a high heuristic probability of finding our desired  $J$  with an odd norm  $d$  smaller than  $2\sqrt{p}$ , thereby yielding the corresponding isogeny  $\sigma : E_2 \rightarrow E_1$ . In [21, Appendix B], we provide details on how our method can be adapted to function with even  $d$  as well. The other requirement is that  $I_{C_a} J$  should not be divisible by  $q$ . This is to avoid that the final response  $\kappa = \sigma \circ \varphi$  is divisible by  $q$ , which would imply that  $\kappa$  is independent of the challenge  $C_a$ . In practice, when  $E_1$  is sampled honestly, the probability that  $I_{C_a} J$  is divisible by  $q$  is at about  $q^{-2} \approx 2^{-2\lambda}$ . Hence an ideal  $J$  that satisfies the previous requirements will satisfy this one as well.

With knowledge of  $d$ , the objective now shifts to constructing an auxiliary isogeny  $\delta : E_A \rightarrow E_\delta$  of degree  $2^\alpha - qd$ . This specific mechanism lies at the heart of SQIPrime2D and underscores the necessity for the secret isogeny  $\tau$  to be of degree  $q$ . The approach involves sampling  $\gamma \in \text{End}(E_0)$ , an endomorphism of degree  $d(2^\alpha - qd)$ . This is done using **FullRepresentInteger**. Next, we compute  $\begin{pmatrix} V \\ W \end{pmatrix} = \tau \circ \hat{\gamma} \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix}$ . Given that  $\deg(\tau \circ \hat{\gamma}) = dq(2^\alpha - qd)$ , we find ourselves in the following scenario:



where  $\gamma = \gamma_2 \circ \gamma_1$ ,  $\deg(\gamma_1) = d$  and  $\deg(\gamma_2) = (2^\alpha - qd)$ . By applying Kani's Lemma, we construct the dimension 2 isogeny  $F : E_0 \times E_A \rightarrow E'_0 \times E_\delta$  of kernel  $\ker(F) = \{([-qd]P, \tau \circ \hat{\gamma}(P)) \mid P \in E_0[2^\alpha]\}$  and given by



$$F := \begin{pmatrix} \widehat{\gamma}_2 & -\gamma_1 \circ \widehat{\tau} \\ [\gamma_2]_* (\tau \circ \widehat{\gamma}_1) & [\tau \circ \widehat{\gamma}_1]_* \gamma_2 \end{pmatrix}.$$

We thus have an efficient representation of our desired  $\delta = [\tau \circ \widehat{\gamma}_1]_* \gamma_2$ .

The response to our challenge is to give the evaluation  $T, U$  of  $\delta \circ \widehat{\kappa} = \delta \circ \widehat{\varphi} \circ \widehat{\sigma}$  over a basis of  $E_1[2^\alpha]$  to the verifier. Additionally, we share the image  $V = \delta(C_a)$  of  $C_a$  through  $\delta$ . To do the evaluation, we call **CanonicalTorsionBasis** over  $E_1$  to deterministically find a basis  $X, Y$  of  $E_1[2^\alpha]$ , evaluate  $\widehat{\kappa}$  on  $X$  and  $Y$  using the **EvalTorsion** and compute  $\delta$  on these images using the dimension two isogeny  $F$ . Finally, we multiply the final points by  $(-qd)^{-1} \pmod{2^\alpha}$ . The prover then sends these three points together with the curve  $E_\delta$ .

---

**Algorithm 10. SQIPrime2D.Response**

---

**Input:**  $\text{pp} = (p, \alpha, q, N, (P_0, Q_0), (P, Q, \iota, I_P))$ ,  $\text{sk} = (\tau, \widehat{\rho}, I_\tau, I_{\widehat{\rho}}, \mathbf{M})$ ,  $\text{pk} = (E_A, (R, S))$ ,  $\text{sec} = (\psi, I_\psi)$ ,  $\text{com} = E_1$ ,  $\text{chal} = a$ .

**Output:**  $\text{res} = (E_\delta, T, U, V)$  with  $T, U \in E_\delta[2^\alpha]$ .

- 1:  $\begin{pmatrix} b \\ c \end{pmatrix} \leftarrow \mathbf{M}^\top \begin{pmatrix} 1 \\ a \end{pmatrix}$
  - 2:  $I_{C_a} \leftarrow [(b + ct)I_\tau]_* I_P$
  - 3:  $J \leftarrow \mathbf{RandomEquivalentIdeal}(\overline{I_{C_a} I_\tau I_\psi}) \quad d \leftarrow n(J)$
  - 4: If  $2|d$  or  $I_{C_a} J$  is divisible by  $q$ , go back to step 3.
  - 5:  $X, Y \leftarrow \mathbf{CanonicalTorsionBasis}(E_1, 2^\alpha)$
  - 6:  $\gamma \leftarrow \mathbf{FullRepresentInteger}(\mathfrak{D}_0, d(2^\alpha - dq))$
  - 7:  $\begin{pmatrix} V \\ W \end{pmatrix} = \tau \circ \widehat{\gamma} \begin{pmatrix} P_0 \\ Q_0 \end{pmatrix}$
  - 8:  $\mathbf{B} \leftarrow \{([-dq]P_0, V), ([-dq]Q_0, W)\}$
  - 9:  $F \leftarrow \mathbf{HDKernelToIsogeny}(E_0 \times E_1, \mathbf{B})$
  - 10: Define  $\tau = F_A(-, 0)_1$  and  $\psi = F_1(-, 0)_1$
  - 11:  $T_1, U_1 \leftarrow \mathbf{EvalTorsion}(\mathfrak{D}_0, \tau, I_\tau, \psi, I_\psi, \overline{I_{C_1} J}, qd, \{X, Y\}) \triangleright T_1 = \widehat{\kappa}(X), U_1 = \widehat{\kappa}(Y)$
  - 12:  $\begin{pmatrix} T \\ U \end{pmatrix} = [(-qd)^{-1}] \delta \begin{pmatrix} T_1 \\ U_1 \end{pmatrix} \quad \triangleright \delta(-) = F(0, -)_2$
  - 13:  $V = \delta(R + [a]S)$
  - 14: Recover  $E_\delta$ , the codomain of  $\delta$
  - 15: **return**  $\text{res} = (E_\delta, T, U, V)$
- 

**6.4 SQIPrime2D Verification Algorithm**

Note that in SQIPrime2D, the verifier receives a dimension 2 representation of  $\widehat{\kappa}$  rather than that of  $\kappa$ . We describe how to use this representation of  $\widehat{\kappa}$  to effectively check that  $\kappa$  is an isogeny from  $E_A$  to  $E_1$  such that  $\ker(\kappa)[q] = \langle C_a \rangle$ .

Upon receipt of  $T, U$  and  $V$ , the verifier deterministically computes the basis  $\langle X, Y \rangle = E_1[2^\alpha]$ . Following that, the verifier uses  $X, Y, T$  and  $U$  to compute a basis for the kernel of the isogeny  $F$ , as derived from Kani’s Lemma over the

following diagram.

$$\begin{array}{ccc}
 E_A & \xrightarrow{\kappa} & E_1 \\
 \delta \downarrow & \delta \circ \widehat{\kappa} \swarrow & \downarrow \kappa_* \delta \\
 E_\delta & \xrightarrow{\delta_* \kappa} & E_\bullet
 \end{array}$$

$$F : E_1 \times E_\delta \rightarrow E_A \times E_\bullet \text{ is defined as } \begin{pmatrix} \widehat{\kappa} & -\widehat{\delta} \\ \kappa_* \delta & \delta_* \kappa \end{pmatrix}$$

$$\ker(F) = \langle \langle [-qd]X, \delta \circ \widehat{\kappa}(X) \rangle, \langle [-qd]Y, \delta \circ \widehat{\kappa}(Y) \rangle \rangle = \langle \langle (X, T), (Y, U) \rangle \rangle$$

Using  $F$ , they compute the point  $F(V) = \begin{pmatrix} -\widehat{\delta}(V) \\ \delta_* \kappa(V) \end{pmatrix}$  and check that:

1.  $\delta_* \kappa(V) = 0$ .
2.  $\widehat{\delta}(V) = [2^\alpha - qd](R + [a]S) = [2^\alpha](R + [a]S)$ .

Additionally, we check that for  $W \in E_\delta[q]$  linearly independent with  $V$ ,  $\delta_* \kappa(W) \neq 0$ . This ensures that  $\ker(\kappa)[q] = \langle C_a \rangle$ .

---

**Algorithm 11. SQIPrime2D.Verify**

---

**Input:**  $\text{pp} = (p, (P_0, Q_0), (P, Q, \iota, I_P))$ ,  $\text{pk} = (E_A, R, S)$ ,  $\text{com} = (E_1)$ ,  $\text{chal} = a$ ,  $\text{res} = (E_\delta, T, U, V)$ .

**Output:** 0 or 1.

- 1: Check  $T, U, V \in E_\delta$
  - 2:  $X, Y \leftarrow \mathbf{CanonicalTorsionBasis}(E_1, 2^\alpha)$
  - 3:  $\mathbf{B} \leftarrow \{(X, T), (Y, U)\}$
  - 4:  $F \leftarrow \mathbf{HDKernelToIsogeny}(E_1 \times E_\delta, \mathbf{B})$  ▷ If not well defined, return 0
  - 5: **if**  $\text{codomain } \widehat{\kappa} \neq E_A$  **do return 0**
  - 6: Sample  $W \in E_\delta[q]$  such that  $V$  and  $W$  are linearly independent
  - 7:  $\begin{pmatrix} Z_1 \\ Z_2 \end{pmatrix} \leftarrow F(V) = \begin{pmatrix} -\widehat{\delta}(V) \\ \delta_* \kappa(V) \end{pmatrix}$
  - 8:  $b_1 \leftarrow Z_1 \stackrel{?}{=} [2^\alpha](R + [a]S)$
  - 9:  $b_2 \leftarrow Z_2 \stackrel{?}{=} 0$
  - 10:  $b_3 \leftarrow \delta_* \kappa(W) \stackrel{?}{\neq} 0$
  - 11: **return**  $b_1 \wedge b_2 \wedge b_3$
- 

The following proposition shows us that our verification is correct.

**Proposition 4.** *Let  $\text{pp}, \text{pk}, \text{com}, \text{chal}$  be the public parameters, a valid public key, a commitment, and a challenge in SQIPrime2D and let  $X, Y$  be the canonical basis of  $E_1[2^\alpha]$ . Let  $\overline{\text{Res}} = (\overline{E}_\delta, \overline{T}, \overline{U}, \overline{V})$  be any possible output of Algorithm 10.*

*If  $\mathbf{SQIPrime2D.Verify}(\text{pp}, \text{pk}, \text{com}, \text{chal}, \overline{\text{Res}}) = 1$ , then  $(X, Y, \overline{T}, \overline{U})$  is a dim 2 representation of an isogeny  $\widehat{\kappa} : E_1 \rightarrow E_A$  of degree  $q\overline{d} < 2^\alpha$  and such that  $\widehat{\kappa}$  factors through  $\varphi$ , the isogeny corresponding to the challenge  $\text{chal}$ , but is not divisible by  $q$ ; in other words,  $\ker(\widehat{\kappa})[q] = \langle C_a \rangle$ .*

*Proof.* Let  $\overline{E}_\delta, \overline{T}, \overline{U}, \overline{V}$  be an accepting response. Since the  $(2^\alpha, 2^\alpha)$  isogeny  $\overline{F}$  whose kernel is generated by  $\{(X, \overline{T}), (Y, \overline{U})\}$  is well-defined, then  $\deg(\overline{F}_{1,1}) = \deg(\overline{F}_{2,2})$ ,  $\deg(\overline{F}_{1,2}) = \deg(\overline{F}_{2,1})$  and  $\deg(\overline{F}_{1,1}) + \deg(\overline{F}_{1,2}) = 2^\alpha$ .

Thus, as  $\overline{F}_{2,2}(\overline{V}) = 0$ , we know that  $q$  divides  $\deg(\overline{F}_{2,2})$ , meaning that it cannot divide  $\deg(\overline{F}_{1,2})$ . Since  $\overline{F}_{1,2}(\overline{V}) = [2^\alpha](R + [a]S)$ , then  $[2^\alpha]\widehat{\overline{F}}_{1,2}(R + [a]S) = [\deg(\overline{F}_{1,2})]\overline{V}$ . As  $q$  and  $2^\alpha \deg(\overline{F}_{1,2})$  are co-prime, we have that  $\widehat{\overline{F}}_{2,2} \circ \widehat{\overline{F}}_{1,2}(R + [a]S) = 0 = \widehat{\overline{F}}_{2,1} \circ \widehat{\overline{F}}_{1,1}(R + [a]S)$ . As  $\deg(\overline{F}_{1,2}) = \deg(\overline{F}_{2,1})$  is not divisible by  $q$ , then  $\widehat{\overline{F}}_{1,1}(R + [a]S) = 0$ . We therefore have that  $\widehat{\overline{F}}_{1,1} : E_A \rightarrow E_1$  is of degree  $q\overline{d} < 2^\alpha$  and it factors through the isogeny  $\varphi$  corresponding to the challenge  $\text{chal}$ . Since  $W \in E_\delta[q]$  is such that  $\overline{V}$  and  $W$  are linearly independent, then  $\overline{F}_{2,2}(W) \neq 0$  induces that  $\widehat{\overline{F}}_{2,2}$  and  $\widehat{\overline{F}}_{1,1} : E_A \rightarrow E_1$  are not divisible by  $q$ .  $\square$

## 7 Security Analysis of SQIPrime2D

Similarly to SQIPrime4D, we have to show that SQIPrime2D defines a  $\Sigma$  protocol. We thus have to prove that we have special soundness and are Honest Verifier Zero Knowledge. Our proof of special soundness differs slightly from Proposition 2, as it leverages the primality of  $q$  to address the case where  $\deg(\sigma)$  is not necessarily co-prime to  $q$ .

**Proposition 5.** *Let  $(E_1, \text{chal}_1, T_1, U_1, V_1)$  and  $(E_1, \text{chal}_2, T_2, U_2, V_2)$  be 2 transcripts of SQIPrime2D with identical commitment  $E_1$  and  $\text{chal}_1 \neq \text{chal}_2$ . There exists an extractor  $\mathcal{E}$  that, given both transcripts, can efficiently solve the one endomorphism problem (Problem 1) over  $E_A$ , i.e. find  $\theta_A \in \text{End}(E_A)$  a non-trivial endomorphism.*

*Proof.* Similarly to Proposition 2, we construct  $\theta_A = \widehat{\kappa}_2 \circ \kappa_1 \in \text{End}(E_A)$ . We now show that  $\theta_A$  is non-scalar.

Let  $d_1 = \deg(\sigma_1)$  and  $d_2 = \deg(\sigma_2)$ . Recall that  $d_1q < 2^\alpha < p$ ,  $d_2q < 2^\alpha < p$  and  $q$  is prime. If both are co-prime to  $q$ , then one follows the same reasoning as in the proof of Proposition 2. Let us assume that  $q$  divides  $d_1$  and let  $d_1 = d'_1q$ . Then  $d'_1$  is co-prime to  $q$ , as otherwise, we would have  $d_1q = d''_1q^3 \geq q^3 > 2^\alpha \geq d_1q$  where  $d'_1 = d''_1q$ , leading to a contradiction.

Now, suppose  $\theta_A = [\chi]$ . Since  $\chi^2 = \deg[\chi] = \deg \theta_A = q^2 d_1 d_2 = q^3 d'_1 d_2$ , then  $d_2 = d'_2 q$  with  $d'_2$  co-prime to  $q$ . Hence  $\deg \kappa_1 = q^2 d'_1$  and  $\deg \kappa_2 = q^2 d'_2$  where  $d'_1$  and  $d'_2$  are co-prime with  $q$ . Write  $\kappa_1 = \phi_1 \circ \kappa'_1$  and  $\kappa_2 = \phi_2 \circ \kappa'_2$  where the isogenies  $\kappa'_1, \kappa'_2, \phi_1$  and  $\phi_2$  have degree  $q^2, q^2, d'_1$  and  $d'_2$  respectively. Since  $\theta_A = \widehat{\kappa}_2 \circ \kappa_1$  is a scalar endomorphism and,  $\kappa_1$  and  $\kappa_2$  are not divisible by  $q$  (which is prime), then  $\kappa'_1 = \kappa'_2$ . This implies that  $\ker(\varphi_1) := \ker(\kappa_1) \cap E_A[q] = \ker(\kappa'_1) \cap E_A[q] = \ker(\kappa'_2) \cap E_A[q] = \ker(\kappa_2) \cap E_A[q] =: \ker(\varphi_2)$ , i.e.  $\text{chal}_1 = \text{chal}_2$ , which is a contradiction.  $\square$

Regarding HVZK, there are several differences between SQIPrime4D and SQIPrime2D:

1. We have access to an auxiliary isogeny  $\delta : E_A \rightarrow E_\delta$ .
2. Our isogeny  $\kappa$  is of degree  $qd$  where the requirements that  $d$  is co-prime to  $q$  and  $2^\beta - qd$  is prime congruent to 1 modulo 4 are relaxed.

We therefore need to define our HVZK under new oracles, defined as such.

**Definition 3.** *The Random Uniform Constrained Odd Degree Isogeny Oracle (RUCODIO) is an oracle that takes as input a supersingular curve  $E$  together with  $P \in E[q]$  and returns an efficient representation of an isogeny  $\kappa : E \rightarrow E'$  of degree  $q\ell$  such that:*

- $E'$  is uniformly distributed.
- $\kappa$  is uniformly distributed among all isogenies between  $E$  and  $E'$  such that:
  - $\ell$  is odd with  $q\ell \leq 2^\alpha$ .
  - $\kappa$  is such that  $\kappa(P) = 0$ .

**Definition 4.** *The Auxiliary Isogeny Oracle (AIO) is an oracle that takes as input a supersingular curve  $E$  together with an odd integer  $\ell < 2^\alpha/q$  and returns an efficient representation of an isogeny  $\delta : E \rightarrow E''$  of degree  $2^\alpha - q\ell$  such that it has the same distribution as the auxiliary isogeny computed in Algorithm 10.*

Using RUCODIO and AIO, we can now prove our HVZK.

**Proposition 6.** *Given pp, pk and chal, then there exists a simulator  $\mathcal{S}$  with access to a RUCODIO and AIO that simulates transcripts with a distribution that is computationally indistinguishable from the distribution of transcripts of SQIPrime2D, conditioned to chal.*

*Proof.* Given  $E_A$ , we sample  $a \in \mathbb{Z}_q$  and construct  $C = R + [a]S$  call RUCODIO over  $E_A$  and  $C$ , we retrieve an efficient representation of  $\kappa : E_A \rightarrow E_1$ . We compute  $\ell = \deg(\kappa)/q$  and call AIO over  $E_A$  and  $d$  to retrieve  $\delta : E_A \rightarrow E_\delta$ . We use this representation to compute the points  $T = \delta \circ \widehat{\kappa}(X), U = \delta \circ \widehat{\kappa}(Y)$  and  $V = \delta(C)$  with  $X, Y$  the canonical basis over  $E_1[2^\alpha]$ . We then simply return the following transcript  $(E_1, a, E_\delta, T, U, V)$ .

This transcript is computationally indistinguishable from a genuine transcript, as:

- A genuine  $E_1$  or one given by RUCODIO are computationally indistinguishable, following Assumption 1.
- Due to Definition 4,  $E_\delta$  has the same distribution as the isogeny computed during SQIPrime2D response. This also applies to the point  $V$ .
- Following [32, Lemma 3.2.4], a genuine  $\kappa$  or one given by RUCODIO are computationally indistinguishable, and so does  $T, U$ . □

We now make the following assumption.

**Assumption 4.** *The one endomorphism problem (Problem 1) remains hard even when given access to RUCODIO and AIO.*

Thus, we have that, under our assumptions, SQIPrime2D is a  $\Sigma$ -protocol.

## 8 Parameters and Efficiency

As discussed in Sect. 4 and Sect. 6, the public parameters in both versions of SQIPrime differ significantly from those used in SQISign [18, 19] and SQISignHD [11], particularly concerning their base prime numbers. This section provides a detailed explanation on how to compute suitable baseline “SQIPrime-friendly” primes.

### 8.1 Finding “SQIPrime4D-Friendly” Primes

We can view “SQIPrime4D-friendly” primes as a combination of the “SIDH primes” used in SQISignHD and the stringent requirements on both  $p + 1$  and  $p - 1$  seen in SQISign primes. However, in SQIPrime4D, the only condition is that  $p - 1$  needs to have a factor of size  $O(2^\lambda)$ . Finding “SQIPrime4D-friendly” primes is actually easier than finding “SQISign-friendly” primes. These primes can in fact be found by a brute-force search over the cofactor  $f$ . Here are some good candidates.

$$\begin{aligned} \lambda = 128 : p + 1 &= 2^{241} \cdot 33967 \simeq 2^{256} \\ q &= 647133889352330391744288229376113975777 \simeq 2^{128} \end{aligned}$$

$$\begin{aligned} \lambda = 192 : p + 1 &= 2^{368} \cdot 239 \cdot 277 \simeq 2^{384} \\ q &= 3 \cdot 7 \cdot 4803463386334137403 \cdot \\ &116682096886878909945888202135243873061 \simeq 2^{193} \end{aligned}$$

$$\begin{aligned} \lambda = 256 : p + 1 &= 2^{497} \cdot 5^2 \cdot 479 \simeq 2^{512} \\ q &= 97 \cdot 147869462015622684206054234380684709202350 \\ &1415545736430515280986935609000677 \simeq 2^{256} \end{aligned}$$

### 8.2 Finding “SQIPrime2D-Friendly” Primes

Finding “SQIPrime2D-friendly” primes through a brute-force search over the cofactor  $f$  as we did in the case of SQIPrime4D is computationally involved. This essentially comes from the fact that we want  $q \approx 2^\lambda$  to be prime this time and if we take  $p = 2^\alpha f - 1$  to be a prime, then the probability that a random prime  $q$  divides  $p - 1$  is roughly  $1/q$ . Given that there are approximately  $2^\lambda(2^t - 1)/\lambda$  distinct primes in the interval  $[2^\lambda, 2^{\lambda+t}]$ , the probability that there exists a prime  $q$  in  $[2^\lambda, 2^{\lambda+t}]$  that divides  $p - 1$  is heuristically given by:

$$\begin{aligned} \mathbb{P} \left[ \exists q \in [2^\lambda, 2^{\lambda+t}] \text{ such that } q \mid (p - 1) \right] &\geq \sum_{q \geq 2^\lambda}^{2^{\lambda+t}} \mathbb{P}[q \mid (p - 1)] \simeq \sum_{q \geq 2^\lambda}^{2^{\lambda+t}} \frac{1}{q} \\ &\geq \sum_{i=1}^t \sum_{q \geq 2^{\lambda+i-1}}^{2^{\lambda+i}} \frac{1}{2^{\lambda+i}} \simeq \sum_{i=1}^t \frac{2^{\lambda+i}}{(\lambda + i)} \frac{1}{2^{\lambda+i}} \simeq \sum_{i=1}^t \frac{1}{\lambda + i} \geq \frac{t}{\lambda + t} \end{aligned}$$

Following this computation, the probability that, for a given  $f$ ,  $p = 2^\alpha f - 1$  is prime *and*  $p - 1$  has a factor close to  $\lambda$ -bit long is about  $O(1/\lambda^2)$ . This induces that the expected size of  $f$  is around  $2 \log_2(\lambda)$ , meaning that a “SQIPrime2D-friendly” prime for the security level  $\lambda$  is of expected size  $2\lambda + 4 \log_2(\lambda)$  bits, or a little bit larger. These additional  $4 \log_2(\lambda)$  bits present a challenge. For  $\lambda = 128$ , this results in an overhead of approximately 28 bits, which translates to an 11% increase in the size of the base prime  $p$ .<sup>6</sup>

For  $\lambda = 128$ , the first “SQIPrime2D-friendly” prime we identified is denoted as  $p_{130}$ .

$$\begin{aligned} p_{130} &= 2^{273} \cdot 19^2 - 1 \simeq 2^{281.50} \\ q_{130} &= 1733124013302036320718171822563477047667 \simeq 2^{130.35} \end{aligned}$$

To find a smaller  $p$ , it is tempting to ask for  $q$  to be non-prime, as we did for SQIPrime4D, but this is not possible as our security would be downgraded by a non-smooth generalisation of the Galbraith meet-in-the-middle attack [26] and our proof of special soundness would be affected. However, to maintain efficiency, we may tolerate a slight reduction in the bit length of  $q$ . We suggest the following prime.

$$\begin{aligned} p_{117} &= 2^{247} \cdot 79 - 1 \simeq 2^{253.34} \\ q_{117} &= 168118140144706967996895604212334429 \simeq 2^{117.01} \end{aligned}$$

Searching for “SQIPrime2D-friendly” primes corresponding to security levels  $\lambda = 192, 256$ , by brute-force search over the cofactor  $f$ , is practically out of reach since it requires factoring several numbers of about 384 and 512 bits. We therefore use a more advanced method which consists of sample integers  $p = 2x^2 - 1$  where  $x = 2^r f_0$  with  $r > \lambda$  and  $f_0$  being a small integer. When  $p$  is prime, then since  $p - 1 = 2x^2 - 2 = 2(x - 1)(x + 1)$ , we only need to check whether  $x + 1$  or  $x - 1$  has a prime  $q \approx 2^\lambda$ . Interestingly, this essentially reduces to checking whether  $x + 1$  or  $x - 1$  has a small smooth factor  $s$  in the order of  $x/2^\lambda$  such that  $(x - 1)/s$  or  $(x + 1)/s$  is prime. This hence leads to a quite efficient method to generate “SQIPrime2D-friendly” primes. A similar technique [7] was also used in the context of SQISign for the parameter generation. We obtained the following primes for the security levels  $\lambda = 192, 256$  respectively.

$$\begin{array}{l|l} p_{186} = 2^{397} \cdot 3^2 \cdot 7^2 \cdot 11^2 - 1 \simeq 2^{413} & p_{240} = 2^{499} \cdot 3^2 \cdot 7^2 - 1 \simeq 2^{508} \\ q_{186} = (2^{198} \cdot 3 \cdot 7 \cdot 11 - 1)/664723 \simeq 2^{187} & q_{240} = (2^{249} \cdot 3 \cdot 7 - 1)/7709 \simeq 2^{241} \end{array}$$

<sup>6</sup> It is important to note that this overhead scales logarithmically with  $\lambda$ . As  $\lambda$  doubles, the overhead only increases by 4 bits, meaning that its relative cost decreases at higher security levels.

### 8.3 Compactness of SQIPrime

Similarly to SQISign and SQISignHD, both version of SQIPrime are made into digital signature schemes via the Fiat-Shamir transform [23]. Thoses digital schemes are universally unforgeable under chosen message attacks (UU-CMA) in the random oracle and RUCGDIO or RUCODIO+AIO model, assuming the hardness of the one endomorphism problem.

**Signature Size.** In the case of SQIPrime2D, the signature takes the form  $\text{sign} = (E_1, E_\delta, T, U, V)$ . This signature can be slightly compressed using methods akin to those outlined in [11, Section 6.1]. The crux of this compression lies in representing  $T$  and  $U$  by  $a_1, a_2, a_3, \in \mathbb{Z}_{2^\alpha}$  corresponding to their coordinates in a deterministic basis of  $E_\delta[2^\alpha]$ , with the final coordinate  $a_4$  derived using pairings and discrete logs and using  $d$  an integer of  $\lambda$  bits. Employing this compression method, each component of a SQIPrime2D signature exhibits the following sizes:

- $E_1$  and  $E_\delta$  are represented by their  $j$ -invariant in  $\mathbb{F}_{p^2}$ , hence of size  $8\lambda + O(\log \lambda)$ .
- $T$  and  $U$  are each represented by three integers of size  $\alpha$  plus  $d$  of size  $\log(p)/2$ , totaling  $7\lambda + O(\log \lambda)$  bits.
- Finally, because  $q$  is non-smooth, we can not compress  $V$ , meaning that they are represented as a point in  $\mathbb{F}_{p^2}$ , hence of size  $4\lambda + O(\log \lambda)$ .

Summing these sizes, a SQIPrime2D signature is  $19\lambda + O(\log \lambda)$  bits long. Consequently, it is larger than the signature of SQISignHD, which was  $13/2\lambda + O(\log \lambda)$  bits, and also larger than SQISign, which is at least  $17/2\lambda + O(\log \lambda)$  bits. Nevertheless, it remains a highly compact post-quantum signature scheme.

It is noteworthy that similar compression techniques can be applied to the SQIPrime4D signature, which is of the form  $(E_1, T, U, V, d)$ , resulting in a signature size of  $12\lambda + O(\log \lambda)$  bits. This difference of  $7\lambda$  bits comes from the fact that in SQIPrime2D, we have to share  $E_\delta$  and because in SQIPrime4D, we split the verification in 2 dimension 4 isogenies, therefore only requiring  $2^\beta$ -torsion points, as opposed to  $2^\alpha$  in SQIPrime2D.

### 8.4 SQIPrime Efficiency

The next phase for SQIPrime involves developing an efficient implementation of SQIPrime2D. Building upon the advancements made in [12], as well as leveraging the efficient implementations of SQISign [18,19] and SQISignHD [11], we anticipate that SQIPrime2D will demonstrate very competitive performance. As for now, we have a proof of concept code written in SageMath, with which the key generation, signature and verification take about 720ms, 1750ms and 200ms respectively when using  $p_{117}$ . We will provide a link to the code in the full version [21] of the paper once the code is ready to share, as there is still room for improvement through basic optimisations.

**Table 1.** Size (in bytes) comparison between the different SQI-protocols for public keys and signatures in both normal and compressed form.

Scheme	$\lambda$	pk	signature	signature (compressed)
SQISign	128	64	322	177
	192	92	-	267
	256	128	-	335
SQISignHD	128	64	208	109
	192	92	312	156
	256	128	416	208
SQIPrime4D	128	192	272	240
	192	288	408	288
	256	384	544	384
SQIPrime2D	128	191	320	299
	192	288	517	484
	256	384	635	600

With an advanced implementation, we expect signature time in SQIPrime2D to be relatively slower compared to that of SQISignHD (this is due to the extra cost of generating the auxiliary isogeny in SQIPrime2D when computing the signature), while having a faster verification. This intuition follows from the number of  $(2, 2)$  isogenies required to perform SQIPrime2D, as detailed in [21, Appendix C].

## 9 Conclusion

In this paper, we have designed SQIPrime, an elegant variant of SQISignHD that uses non-smooth degree challenge isogenies. Moreover, we have described a variant, SQIPrime2D, that uses only dimension two isogenies.

We provide a theoretical performance analysis of our schemes and anticipate SQIPrime2D being more efficient compared to SQISignHD. An effective implementation of SQIPrime2D, which should be expected in the near future, will allow us to have a more practical comparison between SQIPrime4D and SQISignHD on one hand, and, SQIPrime2D, SQISign2D-West [2] and SQISign2D-East [37] on the other hand.

**Acknowledgments.** We are grateful to the anonymous ASIACRYPT 2024 reviewers for their useful comments that helped improve this paper. We would like to thank the authors of SQISign2D-West and the authors of SQISign2D-East for useful discussions regarding this topic. We would also like to thank Michael Meyer, Lorenz Panny and Bruno Sterner for describing a fast method to generate the primes  $p$  used in SQIPrime2D.



## References

1. Basso, A.: POKE: A framework for efficient PKEs, split KEMs, and OPRFs from higher-dimensional isogenies. Cryptology ePrint Archive, Paper 2024/624 (2024), <https://eprint.iacr.org/2024/624>
2. Basso, A., Feo, L.D., Dartois, P., Leroux, A., Maino, L., Pope, G., Robert, D., Wesolowski, B.: SQIsign2D-West: The Fast, the Small, and the Safer. Cryptology ePrint Archive, Paper 2024/760 (2024), <https://eprint.iacr.org/2024/760>
3. Basso, A., Maino, L., Pope, G.: FESTA: Fast encryption from supersingular torsion attacks. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part VII. LNCS, vol. 14444, pp. 98–126. Springer, Singapore (Dec 2023). [https://doi.org/10.1007/978-981-99-8739-9\\_4](https://doi.org/10.1007/978-981-99-8739-9_4)
4. Bernstein, D.J., De Feo, L., Leroux, A., Smith, B.: Faster computation of isogenies of large prime degree. Open Book Series 4(1), 39–55 (2020)
5. Bernstein, D.J., Hamburg, M., Krasnova, A., Lange, T.: Elligator: elliptic-curve points indistinguishable from uniform random strings. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 967–980. ACM Press (Nov 2013). <https://doi.org/10.1145/2508859.2516734>
6. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 227–247. Springer, Cham (Dec 2019). [https://doi.org/10.1007/978-3-030-34578-5\\_9](https://doi.org/10.1007/978-3-030-34578-5_9)
7. Bruno, G., Santos, M.C.R., Costello, C., Eriksen, J.K., Meyer, M., Næhrig, M., Sterner, B.: Cryptographic smooth neighbors. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part VII. LNCS, vol. 14444, pp. 190–221. Springer, Singapore (Dec 2023). [https://doi.org/10.1007/978-981-99-8739-9\\_7](https://doi.org/10.1007/978-981-99-8739-9_7)
8. Castryck, W., Decru, T.: An efficient key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 423–447. Springer, Cham (Apr 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_15](https://doi.org/10.1007/978-3-031-30589-4_15)
9. Chen, M., Leroux, A., Panny, L.: SCALLOP-HD: group action from 2-dimensional isogenies. In: Tang, Q., Teague, V. (eds.) PKC 2024, Part II. LNCS, vol. 14603, pp. 190–216. Springer, Cham (Apr 2024). [https://doi.org/10.1007/978-3-031-57725-3\\_7](https://doi.org/10.1007/978-3-031-57725-3_7)
10. Costello, C.: B-SIDH: Supersingular isogeny Diffie-Hellman using twisted torsion. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 440–463. Springer, Cham (Dec 2020). [https://doi.org/10.1007/978-3-030-64834-3\\_15](https://doi.org/10.1007/978-3-030-64834-3_15)
11. Dartois, P., Leroux, A., Robert, D., Wesolowski, B.: SQIsignHD: New dimensions in cryptography. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part I. LNCS, vol. 14651, pp. 3–32. Springer, Cham (May 2024). [https://doi.org/10.1007/978-3-031-58716-0\\_1](https://doi.org/10.1007/978-3-031-58716-0_1)
12. Dartois, P., Maino, L., Pope, G., Robert, D.: An algorithmic approach to  $(2, 2)$ -isogenies in the theta model and applications to isogeny-based cryptography. Cryptology ePrint Archive, Paper 2023/1747 (2023), <https://eprint.iacr.org/2023/1747>
13. De Feo, L.: Mathematics of isogeny based cryptography. arXiv preprint [arXiv:1711.04062](https://arxiv.org/abs/1711.04062) (2017)
14. De Feo, L., Delpèch de Saint Guilhem, C., Fouotsa, T.B., Kutas, P., Leroux, A., Petit, C., Silva, J., Wesolowski, B.: Seta: Supersingular encryption from torsion attacks. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 249–278. Springer, Cham (Dec 2021). [https://doi.org/10.1007/978-3-030-92068-5\\_9](https://doi.org/10.1007/978-3-030-92068-5_9)

15. De Feo, L., Fouotsa, T.B., Panny, L.: Isogeny problems with level structure. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VII. LNCS, vol. 14657, pp. 181–204. Springer, Cham (May 2024). [https://doi.org/10.1007/978-3-031-58754-2\\_7](https://doi.org/10.1007/978-3-031-58754-2_7)
16. De Feo, L., Galbraith, S.D.: SeaSign: Compact isogeny signatures from class group actions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 759–789. Springer, Cham (May 2019). [https://doi.org/10.1007/978-3-030-17659-4\\_26](https://doi.org/10.1007/978-3-030-17659-4_26)
17. De Feo, L., Jao, D., Plüt, J.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. *Journal of Mathematical Cryptology* **8**(3), 209–247 (2014)
18. De Feo, L., Kohel, D., Leroux, A., Petit, C., Wesolowski, B.: SQISign: Compact post-quantum signatures from quaternions and isogenies. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 64–93. Springer, Cham (Dec 2020). [https://doi.org/10.1007/978-3-030-64837-4\\_3](https://doi.org/10.1007/978-3-030-64837-4_3)
19. De Feo, L., Leroux, A., Longa, P., Wesolowski, B.: New algorithms for the deuring correspondence - towards practical and secure SQISign signatures. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 659–690. Springer, Cham (Apr 2023). [https://doi.org/10.1007/978-3-031-30589-4\\_23](https://doi.org/10.1007/978-3-031-30589-4_23)
20. Deuring, M.: Die typen der multiplikatorenringe elliptischer funktionenkörper. In: *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg*. vol. 14, pp. 197–272. Springer Berlin/Heidelberg (1941)
21. Duparc, M., Fouotsa, T.B.: SQIPrime: A dimension 2 variant of SQISignHD with non-smooth challenge isogenies. *Cryptology ePrint Archive*, Paper 2024/773 (2024), <https://eprint.iacr.org/2024/773>
22. Duparc, M., Fouotsa, T.B., Vaudenay, S.: SILBE: an Updatable Public Key Encryption Scheme from Lollipop Attacks. *Cryptology ePrint Archive*, Paper 2024/400 (2024), <https://eprint.iacr.org/2024/400>
23. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO’86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). [https://doi.org/10.1007/3-540-47721-7\\_12](https://doi.org/10.1007/3-540-47721-7_12)
24. Fouotsa, T.B.: A note on the prime in SQISignHD. Online (2024), [https://github.com/BorisFouotsa/BorisFouotsa.github.io/blob/main/files/A\\_note\\_on\\_the\\_prime\\_in\\_SQISignHD.pdf](https://github.com/BorisFouotsa/BorisFouotsa.github.io/blob/main/files/A_note_on_the_prime_in_SQISignHD.pdf)
25. Fouotsa, T.B., Petit, C.: SHealS and HealS: Isogeny-based PKEs from a key validation method for SIDH. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 279–307. Springer, Cham (Dec 2021). [https://doi.org/10.1007/978-3-030-92068-5\\_10](https://doi.org/10.1007/978-3-030-92068-5_10)
26. Galbraith, S.D.: Constructing isogenies between elliptic curves over finite fields. *LMS Journal of Computation and Mathematics* **2**, 118–138 (1999)
27. Galbraith, S.D., Petit, C., Silva, J.: Identification protocols and signature schemes based on supersingular isogeny problems. In: Takagi, T., Peyrin, T. (eds.) ASIACRYPT 2017, Part I. LNCS, vol. 10624, pp. 3–33. Springer, Cham (Dec 2017). [https://doi.org/10.1007/978-3-319-70694-8\\_1](https://doi.org/10.1007/978-3-319-70694-8_1)
28. Jao, D., De Feo, L.: Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. In: *Post-Quantum Cryptography: 4th International Workshop, PQCrypto 2011, Taipei, Taiwan, November 29–December 2, 2011. Proceedings 4*. pp. 19–34. Springer (2011). <https://doi.org/10.1007/978-3-642-25405-5>

29. Kani, E.: The number of curves of genus two with elliptic differentials. *Journal für die reine und angewandte Mathematik* **1997**(485), 93–122 (1997).<https://doi.org/10.1515/1997.485.932>
30. Kohel, D., Lauter, K., Petit, C., Tignol, J.P.: On the quaternion-isogeny path problem. *LMS Journal of Computation and Mathematics* **17**(A), 418–432 (2014)
31. Kunzweiler, S.: Efficient computation of  $(2^n, 2^n)$ -isogenies. *Designs, Codes and Cryptography* **92**(6), 1761–1802 (2024)
32. Leroux, A.: Quaternion Algebra and Isogeny-Based Cryptography. Ph.D. thesis, Ecole doctorale de l'Institut Polytechnique de Paris (2022)
33. Leroux, A.: Verifiable random function from the Deuring correspondence and higher dimensional isogenies. *Cryptology ePrint Archive*, Paper 2023/1251 (2023), <https://eprint.iacr.org/2023/1251>
34. Maino, L., Martindale, C., Panny, L., Pope, G., Wesolowski, B.: A direct key recovery attack on SIDH. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 448–471. Springer, Cham (Apr 2023).[https://doi.org/10.1007/978-3-031-30589-4\\_16](https://doi.org/10.1007/978-3-031-30589-4_16)
35. Moriya, T.: IS-CUBE: An isogeny-based compact KEM using a boxed SIDH diagram. *Cryptology ePrint Archive*, Paper 2023/1506 (2023), <https://eprint.iacr.org/2023/1506>
36. Nakagawa, K., Onuki, H.: QFESTA: Efficient algorithms and parameters for FESTA using quaternion algebras. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part V. LNCS, vol. 14924, pp. 75–106. Springer, Cham (Aug 2024).[https://doi.org/10.1007/978-3-031-68388-6\\_4](https://doi.org/10.1007/978-3-031-68388-6_4)
37. Nakagawa, K., Onuki, H.: SQIsign2D-East: A new signature scheme using 2-dimensional isogenies. *Cryptology ePrint Archive*, Paper 2024/771 (2024), <https://eprint.iacr.org/2024/771>
38. NIST: Post-Quantum Cryptography: Digital Signature Schemes, <https://csrc.nist.gov/projects/pqc-dig-sig/standardization>
39. Onuki, H., Nakagawa, K.: Ideal-to-isogeny algorithm using 2-dimensional isogenies and its application to SQIsign. *Cryptology ePrint Archive*, Paper 2024/778 (2024), <https://eprint.iacr.org/2024/778>
40. Page, A., Robert, D.: Introducing Clapoti(s): Evaluating the isogeny class group action in polynomial time. *Cryptology ePrint Archive*, Paper 2023/1766 (2023), <https://eprint.iacr.org/2023/1766>
41. Page, A., Wesolowski, B.: The supersingular endomorphism ring and one endomorphism problems are equivalent. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VI. LNCS, vol. 14656, pp. 388–417. Springer, Cham (May 2024).[https://doi.org/10.1007/978-3-031-58751-1\\_14](https://doi.org/10.1007/978-3-031-58751-1_14)
42. Robert, D.: Fonctions thêta et applications à la cryptographie. Ph.D. thesis, Université Henri Poincaré-Nancy I (2010)
43. Robert, D.: Evaluating isogenies in polylogarithmic time. *Cryptology ePrint Archive*, Paper 2022/1068 (2022), <https://eprint.iacr.org/2022/1068>
44. Robert, D.: Breaking SIDH in polynomial time. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 472–503. Springer, Cham (Apr 2023).[https://doi.org/10.1007/978-3-031-30589-4\\_17](https://doi.org/10.1007/978-3-031-30589-4_17)
45. Santos, M.C.R., Costello, C., Smith, B.: Efficient  $(3,3)$ -isogenies on fast Kummer surfaces. *Cryptology ePrint Archive*, Paper 2024/144 (2024), <https://eprint.iacr.org/2024/144>
46. Santos, M.C.R., Eriksen, J.K., Meyer, M., Reijnders, K.: AprèsSQI: Extra fast verification for SQIsign using extension-field signing. In: Joye, M., Leander, G.

- (eds.) EUROCRYPT 2024, Part I. LNCS, vol. 14651, pp. 63–93. Springer, Cham (May 2024). [https://doi.org/10.1007/978-3-031-58716-0\\_3](https://doi.org/10.1007/978-3-031-58716-0_3)
47. Silverman, J.H.: The arithmetic of elliptic curves, vol. 106. Springer (2009)
48. Vêlu, J.: Isogénies entre courbes elliptiques. *Comptes-Rendus de l'Académie des Sciences* **273**, 238–241 (1971)
49. Yoo, Y., Azarderakhsh, R., Jalali, A., Jao, D., Soukharev, V.: A post-quantum digital signature scheme based on supersingular isogenies. In: Kiayias, A. (ed.) FC 2017. LNCS, vol. 10322, pp. 163–181. Springer, Cham (Apr 2017). [https://doi.org/10.1007/978-3-319-70972-7\\_9](https://doi.org/10.1007/978-3-319-70972-7_9)

# **Post-quantum Cryptography**



# CPA-Secure KEMs are also Sufficient for Post-quantum TLS 1.3

Biming Zhou<sup>1,3</sup>, Haodong Jiang<sup>2(✉)</sup>, and Yunlei Zhao<sup>1,3(✉)</sup>

<sup>1</sup> College of Computer Science and Technology, Fudan University, Shanghai 200433, China

[bmzhou22@m.fudan.edu.cn](mailto:bmzhou22@m.fudan.edu.cn), [ylzhao@fudan.edu.cn](mailto:ylzhao@fudan.edu.cn)

<sup>2</sup> Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, Henan, China

[hdjiang13@163.com](mailto:hdjiang13@163.com)

<sup>3</sup> State Key Laboratory of Cryptology, Beijing 100878, China

**Abstract.** In the post-quantum migration of TLS 1.3, an ephemeral Diffie-Hellman must be replaced with a post-quantum key encapsulation mechanism (KEM). At EUROCRYPT 2022, Huguenin-Dumittan and Vaudenay [20] demonstrated that KEMs with standard CPA security are sufficient for the security of the TLS 1.3 handshake. However, their result is only proven in the random oracle model (ROM), and as the authors comment, their reduction is very much non-tight and not sufficient to guarantee security in practice due to the  $O(q^6)$ -loss, where  $q$  is the number of adversary's queries to random oracles. Moreover, in order to analyze the post-quantum security of TLS 1.3 handshake with a KEM, it is necessary to consider the security in the quantum ROM (QROM). Therefore, they leave the tightness improvement of their ROM proof and the QROM proof of such a result as an interesting open question.

In this paper, we resolve this problem. We improve the ROM proof in [20] from an  $O(q^6)$ -loss to an  $O(q)$ -loss with standard CPA-secure KEMs which can be directly obtained from the underlying public-key encryption (PKE) scheme in CRYSTALS-Kyber [8]. Moreover, we show that if the KEMs are constructed from rigid deterministic public-key encryption (PKE) schemes such as the ones in Classic McEliece [2] and NTRU [11], this  $O(q)$ -loss can be further improved to an  $O(1)$ -loss. Hence, our reductions are sufficient to guarantee security in practice. According to our results, a CPA-secure KEM (which is more concise and efficient than the currently used CCA/1CCA-secure KEM) can be directly employed to construct a post-quantum TLS 1.3. Furthermore, we lift our ROM result into QROM and first prove that the CPA-secure KEMs are also sufficient for the post-quantum TLS 1.3 handshake. In particular, the techniques introduced to improve reduction tightness in this paper may be of independent interest.

**Keywords:** TLS 1.3 · tightness · quantum random oracle model · KEM-TLS

## 1 Introduction

The Transport Layer Security (TLS) protocol is one of the most widely deployed cryptographic protocols in practice. As the NIST standardization of post-quantum cryptography (PQC) progresses, exploring the transition of the TLS 1.3 protocol to post-quantum (PQ) security has become a significant topic. To ensure PQ security for parts of the protocol that use Diffie-Hellman (DH) key exchange, it is necessary to replace the existing DH key exchange with a PQ secure key encapsulation mechanism (KEM).

The existing TLS 1.3 protocol [15], as well as the majority of other cryptographic protocols such as KEM-TLS [30], Signal [9], and Noise [4] schemes, rely on the PRF-ODH [10] assumption to achieve security. The PRF-ODH assumption is a variant of the hashed DH assumption. For PQ variants of these protocols, it has been shown that IND-1CCA security is required for the replaced KEMs, see PQ TLS [15, 20, 30, 31], PQ Signal [9], and PQ Noise [4]. Simply put, IND-1CCA security ensures that any probabilistic polynomial time (PPT) adversary cannot distinguish between a legitimately generated key and a random key with at most one decapsulation query. Usually, IND-CCA secure KEMs are taken as IND-1CCA secure KEMs for the implementation of PQ TLS 1.3 [1]. IND-CCA security is typically achieved by employing a Fujisaki-Okamoto-like (FO-like) transformation to an OW-CPA/IND-CPA secure public-key encryption (PKE), see [7, 14, 16–19, 22–25]. In particular, CRYSTALS-Kyber [8] and the remaining KEMs in the Round-4 submissions [27] all employ an FO-like transformation. However, the FO-like transformation in IND-CCA secure KEMs requires re-encryption in decapsulation, significantly impacting decapsulation efficiency as demonstrated in [20]. For instance, there is a 2.17X speedup over decapsulation in CRYSTALS-Kyber [8], and a 6.11X speedup in FrodoKEM [26] when re-encryption is removed, as shown in [20]. Moreover, re-encryption can render KEMs more vulnerable to side-channel attacks as demonstrated in [5, 32], affecting nearly all NIST-PQC Round-3 KEMs. Therefore, the design of an IND-1CCA secure KEM without re-encryption was left as an open problem in [30].

Huguenin-Dumittan and Vaudenay [20] made the first attempt to solve this problem by proposing two general constructions of IND-1CCA secure KEMs from OW-CPA/IND-CPA PKEs. One construction, denoted as  $T_{CH}$ , incorporates a key-confirmation component into the original ciphertext, causing ciphertext expansion. Another construction, denoted as  $T_H$ , works without ciphertext expansion, and the key is derived by  $H(m, c)$ . Building upon [20], Jiang et al. [21] introduced an implicit variant of  $T_H$ , denoted as  $T_{RH}$ , and provided a tighter security reduction for both  $T_H$  and  $T_{RH}$  in the Random Oracle Model (ROM) compared to the proof presented in [20]. Jiang et al. [21] also established the security of  $T_H$  and  $T_{RH}$  in the Quantum Random Oracle Model (QROM) by introducing a variant of the measure-and-reprogram technique [12, 13].

Paquin, Stebila, and Tamvada [29] conjectured that CPA KEMs are sufficient for TLS 1.3. As shown in Fig. 1, the construction of CPA KEMs is more concise than 1CCA KEMs as one hash calculation can be removed. Huguenin-Dumittan and Vaudenay [20] confirmed this conjecture. They observed that in the TLS

1.3 key schedule, the keys are obtained by applying key-derivation functions (KDFs) to the shared secret and the hash of the transcript so far (including the ciphertext). Inspired by the proof of security of the  $T_H$  transform, they proved that if the underlying KEM is OW-CPA secure, then the TLS 1.3 handshake protocol is secure in the MultiStage model of Dowling et al. [15]. Specifically, they introduced a distinct intermediate IND-1CCA-MAC game to demonstrate that OW-CPA KEMs are sufficient for TLS 1.3 in the ROM. They first proved that OW-CPA KEMs imply the security of the IND-1CCA-MAC with a secure MAC in the ROM, then utilized the security of the IND-1CCA-MAC to prove the security of TLS 1.3 in the standard model. Notably, the IND-1CCA-MAC game only serves as an intermediate step in the proof.

However, they only proved that OW-CPA KEMs can derive IND-1CCA-MAC security with a secure MAC in the ROM [20]. They did not extend their proof to the QROM. Also, the bound of the ROM proof is very much non-tight, with  $\epsilon_R \approx O(1/q^6)\epsilon_{\mathcal{A}}$  for OW-CPA KEMs to prove IND-1CCA-MAC secure, where  $\epsilon_R$  (resp.  $\epsilon_{\mathcal{A}}$ ) is the advantage of the reduction  $R$  (resp. adversary  $\mathcal{A}$ ) breaking the OW-CPA security of the underlying KEM (resp. the IND-1CCA-MAC security), and  $q$  is the number of  $\mathcal{A}$ 's queries to the random oracle (RO). Therefore, they suggest that due to the weak security bound associated with OW-CPA KEMs, employing IND-1CCA KEMs might be more advantageous in the PQ TLS 1.3 handshake because the security bound for IND-1CCA KEMs provides better guarantees than OW-CPA KEMs. Consequently, better parameters can be chosen for the implementation of the underlying IND-1CCA KEMs. Furthermore, they leave the development of tighter ROM reductions and proofs in the QROM as open problems, as stated in Sect. 4.3 of their paper [20].

*The overall bound for TLS security from OW-CPA is very much non-tight. This is clearly not sufficient to guarantee security in practice, and we leave the improvement of the bounds as an interesting open question and leave security in the QROM as future work.*

## 1.1 Our Contributions

We resolve the open problem by introducing a new intermediate security game IND-1CCA-MAC\* (a variant of the IND-1CCA-MAC [20] and suitable for establishing the security of TLS 1.3 handshake), and by reducing the CPA security of the underlying KEM to IND-1CCA-MAC\* in a tighter manner. In particular, our results show that standard CPA-secure KEMs are sufficient to guarantee the security of TLS 1.3 in practice. Our main contributions are as follows:

1. First, we prove the security of IND-1CCA-MAC\* from standard CPA-secure KEMs in the ROM. Specifically, our reduction exhibits a tightness of  $\epsilon_R \approx O(1/q)\epsilon_{\mathcal{A}}$ , which is much tighter than  $\epsilon_R \approx O(1/q^6)\epsilon_{\mathcal{A}}$  given by [20] (see Table 1). Such a CPA-secure KEM can be directly obtained by instantiating the PKE scheme in Fig. 1 with the one used in CRYSTALS-Kyber [8].



2. Moreover, we also show that for rigid D-OW-CPA KEMs the reduction can be tight, with  $2\epsilon_R \approx \epsilon_{\mathcal{A}}$ . Here, rigid D-OW-CPA KEMs denote KEMs that are constructed by applying a simple transform to a rigid one-way secure deterministic PKE<sup>1</sup>, as shown in Fig. 1. In particular, the NIST-PQC Round-3 Finalist NTRU [11] and the NIST-PQC Round-4 Candidate Classic McEliece [2] are based on rigid one-way secure deterministic PKEs, which can be transformed into the corresponding D-OW-CPA KEMs in this paper.
3. Then, we first prove the security of IND-1CCA-MAC\* from OW-CPA/IND-CPA/D-OW-CPA<sup>2</sup> KEMs in the QROM. In particular, Huguenin-Dumittan and Vaudenay [20] conjectured that the compressed oracle technique introduced by Zhandry [34] could be useful in the QROM proof due to their extensive use of the programming property of ROs in the ROM proof. However, in our QROM proof, we only utilize two other well-established techniques: one-way to hiding (OW2H) [3, 7] and measure-and-reprogram [12, 13, 21]. Specifically, our reduction achieves a tightness of  $\epsilon_R \approx O(1/q^2)\epsilon_A^2$  in the QROM with IND-CPA/D-OW-CPA secure KEMs.
4. Finally, we show that if the IND-1CCA-MAC\* security is satisfied, then the MultiStage [15] security of TLS 1.3 handshake protocol is satisfied in the standard model. In particular, the reduction for TLS 1.3 from IND-1CCA-MAC\* exhibits the same tightness as the reduction given by [20] for TLS 1.3 from IND-1CCA-MAC or 1CCA KEM. Putting everything together, we finally prove that if the underlying KEM is OW-CPA/IND-CPA/D-OW-CPA secure, then the TLS 1.3 handshake protocol is secure in the MultiStage model with a much tighter ROM proof and the first QROM proof.

*Remark 1.* Our results show that the reduction bounds for IND-1CCA-MAC\* from CPA KEMs exhibit the same tightness as those 1CCA KEMs from PKEs [21]. We also note that CPA KEMs in Fig. 1 can be tightly reduced to CPA PKEs. Therefore, if we consider the complete reduction from the CPA security of the underlying PKE to the MultiStage security of the resulting TLS 1.3, our reduction for TLS 1.3 with CPA KEM has the same tightness as the currently tightest reduction for TLS 1.3 with 1CCA KEM given by [21]. Notably, the CPA-secure KEM used in this paper is more concise and efficient compared to the currently employed CCA/1CCA-secure KEM.

## 1.2 Practical Efficiency Impact

As shown in Fig. 2, the most economical method to construct a secure TLS 1.3 is to use an OW-CPA/IND-CPA secure KEM, obviating the need for transformations like FO or  $T_{CH}, T_H, T_{RH}$  to achieve 1CCA security. Directly using the CPA

<sup>1</sup> The rigid [6] property means that decrypting a ciphertext  $c$  and then re-encrypting yields  $c$ . For a general deterministic PKE, the rigid property can be achieved through a re-encryption transform.

<sup>2</sup> In the QROM, we do not require the rigid property for D-OW-CPA KEMs because the simulations of  $\mathcal{O}^{\text{Dec}}$  and  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  are the same as standard CPA KEMs in the proof.

**Table 1.** Reduction tightness of the intermediate game IND-1CCA-MAC\*.

Underlying KEM	Reduction Tightness <sup>a</sup>	Model
OW-CPA [20]	$\epsilon_R \approx O(1/q^6)\epsilon_A$	ROM
OW-CPA (Our work)	$\epsilon_R \approx O(1/q^2)\epsilon_A$	ROM
IND-CPA (Our work)	$\epsilon_R \approx O(1/q)\epsilon_A$	ROM
D-OW-CPA (Our work)	$\epsilon_R \approx O(1)\epsilon_A$	ROM
OW-CPA [20]	-	QROM
OW-CPA (Our work)	$\epsilon_R \approx O(1/q^4)\epsilon_A^2$	QROM
IND-CPA (Our work)	$\epsilon_R \approx O(1/q^2)\epsilon_A^2$	QROM
D-OW-CPA (Our work)	$\epsilon_R \approx O(1/q^2)\epsilon_A^2$	QROM

<sup>a</sup>All proofs for IND-1CCA-MAC\* from CPA KEMs rely on a secure MAC, we focus on the primary KEM component here. In [20], they actually prove IND-1CCA-MAC, which implies the security of IND-1CCA-MAC\*. Essentially, IND-1CCA-MAC\* is sufficient for the security proof of TLS 1.3, see Theorem 6 for details.

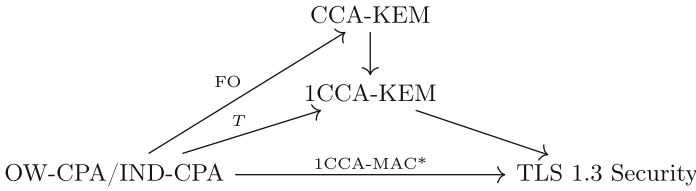
KEM based on CRYSTALS-Kyber.PKE [28] as in Fig. 1 can bring a significant speed improvement, see Table 2. In particular, for decapsulation, there is a 6X speedup over using  $T_{CH}, T_{RH}$ , and a 20X speedup over using FO.

Encaps(pk)	Decaps(sk, c)
1 : $m \leftarrow \mathcal{M}$	1 : $m' = \text{dec}'(\text{sk}, c)$
2 : $c \leftarrow \text{enc}'(\text{pk}, m)$	2 : <b>if</b> $m' = \perp$
3 : $K := m$ //CPA	3 : <b>return</b> $\perp$
4 : $K := H(m, c)$ //1CCA	4 : <b>else return</b> $K := m'$ //CPA
5 : <b>return</b> $(K, c)$	5 : <b>else return</b> $K := H(m', c)$ //1CCA

**Fig. 1.** The construction of CPA and 1CCA ( $T_H$  [20]) KEMs from PKEs

**Table 2.** Benchmark of Encaps and Decaps for CRYSTALS-Kyber [28] with different transforms using liboqs (AVX2 enabled, NIST security level I) on system specs: Intel(R) Core(TM) i9-10900X CPU @ 3.7 GHz, 32.0 GB RAM, 64-bit OS.

Algorithm	CPA	$T_{RH}$	$T_{CH}$	FO
<b>Encaps</b> ( $\mu s$ )	5.35	7.255	7.682	7.666
<b>Decaps</b> ( $\mu s$ )	0.366	2.274	2.277	7.428



**Fig. 2.** Diagram of the process for constructing TLS 1.3 using KEMs based on different security assumptions. This figure shows that the most straightforward method to construct a secure TLS 1.3 is to use an OW-CPA/IND-CPA secure KEM, which does not require the redundant  $T$  transformation [20, 21] or FO transformation [16, 18]. Here,  $T$  represents a general term for  $T_{CH}, T_H, T_{RH}$ .

### 1.3 Technique Overview

**Approach by Huguenin-Dumittan and Vaudenay [20]:** To demonstrate that OW-CPA KEMs are sufficient for the security of TLS1.3 (the description of the TLS 1.3 protocol is illustrated in Fig. 4.), Huguenin-Dumittan and Vaudenay introduced a special intermediate IND-1CCA-MAC game, as depicted in Fig. 3. They first demonstrated that an OW-CPA KEM with a secure MAC implies IND-1CCA-MAC security and then established the security of TLS 1.3 based on the IND-1CCA-MAC security. Specifically, the reduction bounds for deriving TLS 1.3 security from IND-1CCA-MAC KEMs or from IND-1CCA KEMs are both  $O(t_s^2 \epsilon_{\mathcal{A}} + t_s t_u \epsilon_{\mathcal{B}}^{\text{SIG}})$ , where  $t_s$  (resp.  $t_u$ ) is the maximal number of sessions (resp. users) and  $\epsilon_{\mathcal{A}}$  (resp.  $\epsilon_{\mathcal{B}}^{\text{SIG}}$ ) is the advantage of  $\mathcal{A}$  (resp.  $\mathcal{B}$ ) breaking the IND-1CCA/IND-1CCA-MAC security for KEM (resp. EUF-CMA security for signature). The following is an overview of the rationale behind using this special IND-1CCA-MAC game to prove the security of TLS 1.3. When attempting to prove the security of TLS 1.3, it is necessary to simulate the specific entire session and the queries by the adversary. Specifically, when the tested session is the server session labeled  $\text{label}_S$ , and its partner is labeled  $\text{label}_C$ . If the Server Hello (SH) message (including the original SH and Server Key Share (SKS)) sent by  $\text{label}_S$  differs from the SH message received by  $\text{label}_C$ , it indicates potential tampering by the adversary with SH values. However, the reduction must still simulate the honest partner  $\text{label}_C$  to complete the handshake protocol and answer the adversary’s queries. Therefore, by utilizing two oracles,  $\mathcal{O}^{\text{Dec}}$  and  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  in the IND-1CCA-MAC game, the reduction can perfectly simulate this scenario. Specifically, the reduction can query  $\mathcal{O}^{\text{Dec}}$  to obtain stage-1 and stage-2 keys  $tk_C, tk_S$ , enabling perfect simulation of  $\text{label}_C$  and any Reveal queries up to the Server Finished (SF) message. Upon  $\text{label}_C$  receiving the SF message, which contains a MAC tag, the reduction can query  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  to verify this tag. If this tag is correct, the reduction obtains the Handshake Secret (HS) and can derive all necessary secrets to perfectly simulate  $\text{label}_C$ .

**Causes of Reduction Loss in [20]:** When proving that OW-CPA KEMs are IND-1CCA-MAC secure in the ROM, the reduction needs to simulate

$\mathcal{O}^{\text{Dec}}(\overline{ct}, \overline{n})$ ,  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(ct, n, \text{tag}, \text{txt})$  without secret key and embed the underlying security experiment into the IND-1CCA-MAC instance. When simulating  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$ , the reduction randomly takes one of adversary's query  $H_2(\text{HS}, t)$  (corresponding to  $H_2(\text{HS}, H_T(ct, n))$  in  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$ ) or  $\perp$  as the return, with the success probability of guessing correctly being  $O(1/(q_{H_2} + 1))$ . When simulating  $\mathcal{O}^{\text{Dec}}$ , the reduction takes one of the adversary's query  $H_1(\text{HS}, t), H_2(\text{HS}, t)$  (corresponding to  $H_1(\overline{\text{HS}}, H_T(\overline{ct}, \overline{n})), H_2(\overline{\text{HS}}, H_T(\overline{ct}, \overline{n}))$  in  $\mathcal{O}^{\text{Dec}}$ ) or  $\perp$  as the return, or  $\perp_d$  (this guess indicates that the adversary has not queried about the corresponding value  $H_1(\overline{\text{HS}}, H_T(\overline{ct}, \overline{n})), H_2(\overline{\text{HS}}, H_T(\overline{ct}, \overline{n}))$  and  $\text{decaps}(\text{sk}, \overline{ct}) \neq \perp$ ), with the success probability of guessing correctly being  $O(1/(q_{H_1} + 2)(q_{H_2} + 2))$ . In the case of  $\perp_d$ , the reduction randomly chooses  $\overline{chts}, \overline{shts}$ , and finally returns  $H_D(\overline{chts}), H_D(\overline{shts})$  in  $\mathcal{O}^{\text{Dec}}$ . After this, the reduction must ensure the consistency of  $H_1$  and  $H_2$  with  $\mathcal{O}^{\text{Dec}}$  by guessing whether  $H_1(\text{HS}, t), H_2(\text{HS}, t)$  corresponds to the potentially defined  $\overline{chts}, \overline{shts}$  in  $\mathcal{O}^{\text{Dec}}$ , with the success probability of guessing correctly being  $O(1/(q_{H_1} + 1)(q_{H_2} + 1))$ . When embedding the instance of the underlying OW-CPA experiment into the IND-1CCA-MAC instance, an OW-CPA instance is embedded with a  $O(1/q_G)$  loss in the ROM. Thus, the total loss is  $O(1/q^6)$  in the ROM.

Below, we elaborate on how to improve the reduction of the above loss in the ROM and lift our tighter ROM proof into the QROM setting.

**A New Intermediate Game: IND-1CCA-MAC\*:** We observe that we only need a specific IND-1CCA-MAC game denoted as IND-1CCA-MAC\* to prove the security of TLS1.3. IND-1CCA-MAC\* is identical to IND-1CCA-MAC, except it constrains the adversary  $\mathcal{A}$  to initiating the first query to  $\mathcal{O}^{\text{Dec}}(\overline{ct}, \overline{n})$  and subsequent query to  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(ct, n, \text{tag}, \text{txt})$  with  $(ct, n) = (\overline{ct}, \overline{n})$ . This restriction in IND-1CCA-MAC\* does not impact the proof for TLS1.3 because in the proof when the adversary send a forged SH =  $(ct, n_s)$  message the reduction just queries  $\mathcal{O}^{\text{Dec}}(ct, n_s)$  first and  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(ct, n_s, \text{txt}, \text{tag})$  later to perfectly simulate the game. Specifically, following the method in [20], we can easily utilize IND-1CCA-MAC\* to prove the MultiStage security of TLS 1.3. In particular, by utilizing IND-1CCA-MAC\*, we can employ the guess from  $\mathcal{O}^{\text{Dec}}$  to perfectly simulate  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$ , thereby avoiding the  $O(1/(q_{H_2} + 1))$  security loss in the ROM and this new intermediate game also plays a crucial role for the QROM proof.

**Combining RO Simulation Technique:** When there are two ROs,  $H_1(x)$  and  $H_2(x)$ , with identical input space, we need to operate on these two ROs on the same input  $x_0$ , which could be either reprogramming or guessing which query to  $H_1(\cdot), H_2(\cdot)$  corresponds to  $x_0$ . In this paper, we combine these two ROs by defining  $H_{12} = (H_1, H_2)$  to simulate  $H_1$  and  $H_2$  simultaneously. In this scenario, the total number of queries to  $H_{12}$  is at most  $q_{H_1} + q_{H_2}$ . Thus operating only on  $H_{12}$  may result in a tighter reduction loss. More precisely, they have to guess separately for the two ROs, resulting in a loss of  $O(1/(q_{H_1} + 1)(q_{H_2} + 1))$  in [20]. By using the combining RO simulation technique, we only need to make one guess for  $H_{12} = (H_1, H_2)$ , thereby reducing the security loss to  $O(1/(q_{H_1} + q_{H_2} + 1))$ .

Specifically, to simulate the  $\mathcal{O}^{\text{Dec}}(\overline{ct}, \overline{n})$  oracle without the secret key, we initially employ an internal hash function  $H_{12} = (H_1, H_2)$  to simulate the ran-

dom oracles  $H_1$  and  $H_2$ . We first randomly choose  $guess \leftarrow_{\$} \{0, 1\}$  to guess whether  $\text{decaps}(\overline{ct}, \text{sk}) = \perp$ . If  $guess = 0$ , we return  $\perp$  in  $\mathcal{O}^{\text{Dec}}$ . Otherwise, for a valid ciphertext  $\overline{ct}$  such that  $\perp \neq K \leftarrow \text{decaps}(\text{sk}, \overline{ct})$ , the Dec oracle should return  $(H_D(H_1(\overline{\text{HS}}, \overline{t})), H_D(H_2(\overline{\text{HS}}, \overline{t})))$ , where  $\overline{\text{HS}} = G(K)$ ,  $\overline{t} = H_T(\overline{ct}, \overline{n})$ . Consequently, following [21], we directly reprogram  $H_{12}(\overline{\text{HS}}, \overline{t})$  with random values  $\Theta = (\overline{chts}, \overline{shts})$ , and then output  $H_D(\overline{chts})$ ,  $H_D(\overline{shts})$  in the  $\mathcal{O}^{\text{Dec}}$  oracle. Intuitively, the simulation is perfect if we reprogram  $H_{12}(\overline{\text{HS}}, \overline{t})$  with  $\Theta$  when the adversary  $\mathcal{A}$  first queries  $(\overline{\text{HS}}, \overline{t})$  to either  $H_1$  or  $H_2$ . In the ROM, a random guess is correct with a probability of  $1/(q_{H_1} + q_{H_2} + 1)$ .

**Utilizing IND-CPA:** When embedding the underlying security experiment into the IND-1CCA-MAC\* instance, we successfully embed an IND-CPA instance in the ROM without reduction loss, and an OW-CPA instance is embedded in the ROM with an  $O(1/q)$  loss.

**Rigid D-OW-CPA:** We can utilize rigid D-OW-CPA KEMs to prove IND-1CCA-MAC\*/IND-1CCA-MAC tightly. The fundamental reason is that with the rigid property of the underlying deterministic PKE, we can perfectly simulate  $\text{HS} = G(\text{decaps}(\text{sk}, \text{ct}))$  without  $\text{sk}$  when  $\text{decaps}(\text{sk}, \text{ct}) \neq \perp$  if  $G$  is a random oracle. Thus we can use HS perfectly simulate  $\mathcal{O}^{\text{Dec}}$ ,  $\mathcal{O}^{\text{Dec}}_{\text{MAC}}$ . Moreover, we embed a D-OW-CPA instance in the ROM without reduction loss.

Specifically, we note that the  $\delta$ -correctness implies that all queries to  $G(K)$  are keys  $K$  that do not induce correctness errors. When the adversary queries  $G(K)$ , we return HS by lazy sampling and updating  $\mathfrak{L} = \mathfrak{L} \cup (K, \text{ct}, \text{HS})$ , where  $\text{ct} = \text{enc}'(K)$ . Subsequently, when the adversary queries  $\mathcal{O}^{\text{Dec}}(\text{ct}_1, n_1)$  (resp.  $\mathcal{O}^{\text{Dec}}_{\text{MAC}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$ ), we first randomly choose  $guess \leftarrow_{\$} \{0, 1\}$  to guess whether  $\text{decaps}(\text{ct}_1, \text{sk}) = \perp$  (resp.  $\text{decaps}(\text{ct}_2, \text{sk}) = \perp$ ). When  $guess = 0$ , we return  $\perp$ . When  $guess = 1$  and the corresponding  $\text{ct}_1$  (resp.  $\text{ct}_2$ ) exists in  $\mathfrak{L}$ , we directly extract the HS corresponding to  $\text{ct}_1$  (resp.  $\text{ct}_2$ ) from  $\mathfrak{L}$ . Otherwise, we can conclude that the adversary has not yet queried  $G(K)$ , where  $K = \text{dec}'(\text{sk}, \text{ct}_1)$  (resp.  $K = \text{dec}'(\text{sk}, \text{ct}_2)$ ). Assuming the adversary has queried  $G(K)$  before, this implies the existence of  $(K, \text{ct}'_1, \cdot) \in \mathfrak{L}$ , where  $\text{enc}'(\text{pk}, K) = \text{ct}'_1$ . According to the rigid property, we have  $\text{ct}_1 = \text{ct}'_1$ , which contradicts the condition. Therefore, we just directly sample a random HS and record  $\mathfrak{L}_{\text{Dec}} = \{\text{ct}_1, \text{HS}\}$  (resp.  $\mathfrak{L}^{\text{MAC}}_{\text{Dec}} = \{\text{ct}_2, \text{HS}\}$ ). Finally, using HS we can directly simulate  $\mathcal{O}^{\text{Dec}}$  (resp.  $\mathcal{O}^{\text{Dec}}_{\text{MAC}}$ ). To ensure consistency between the random oracle  $G$  and  $\mathcal{O}^{\text{Dec}}$  (resp.  $\mathcal{O}^{\text{Dec}}_{\text{MAC}}$ ), in the simulation of following  $G(K)$ , we first compute  $\text{ct} = \text{enc}'(K)$  and directly return HS from  $\mathfrak{L}_{\text{Dec}}$  (resp.  $\mathfrak{L}^{\text{MAC}}_{\text{Dec}}$ ) if  $\text{ct} = \text{ct}_1$  (resp.  $\text{ct} = \text{ct}_2$ ). If  $\text{ct} \neq \text{ct}_1$  (resp.  $\text{ct} \neq \text{ct}_2$ ), we can assert that  $\text{dec}'(\text{sk}, \text{ct}) \neq \text{dec}'(\text{sk}, \text{ct}_1)$  (resp.  $\text{dec}'(\text{sk}, \text{ct}) \neq \text{dec}'(\text{sk}, \text{ct}_2)$ ) based on the rigid property. Therefore, we can perfectly simulate  $\mathcal{O}^{\text{Dec}}$  and  $\mathcal{O}^{\text{Dec}}_{\text{MAC}}$  when we guess correctly, and the probability of guessing correctly in each case is  $1/2$ . Lastly, the D-OW-CPA adversary can directly identify a  $K$  such that  $\text{enc}'(K) = \text{ct}^*$  in the  $G$ -List and return  $K$  as the solution to the D-OW-CPA instance without random guessing, thereby embedding a D-OW-CPA instance without reduction loss.

**QROM:** We have discussed the primary techniques used to achieve a tighter security reduction for IND-1CCA-MAC\* in the ROM. However, achieving QROM proof for IND-1CCA-MAC\* from OW-CPA/IND-CPA/D-OW-CPA is still challenging. The fundamental difficulty in the proof revolves around embedding the underlying security experiment into the IND-1CCA-MAC\* instance and simulate  $\mathcal{O}^{\text{Dec}}$ ,  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  without the secret key. We delve into how we solve each of these challenges in detail.

To embed the instance of the underlying security experiment into the IND-1CCA-MAC\* instance, if the underlying KEM is OW-CPA secure, we follow previous proofs [7, 21, 22] and use general OW2H [3] to argue the embedding of the underlying instance. Moreover, if the KEM is D-OW-CPA, we employ double-sided OW2H [7] to discuss the implications of instance embedding. Similarly, if the KEM is IND-CPA, we utilize double-sided OW2H and extended double-sided OW2H [21] to discuss the implications of instance embedding.

To simulate the  $\mathcal{O}^{\text{Dec}}(\bar{ct}, \bar{n})$  oracle without the secret key, we initially utilize an internal hash function  $H_{12} = (H_1, H_2)$  to simulate the random oracles  $H_1$  and  $H_2$ . Firstly, we randomly choose  $guess \leftarrow \{0, 1\}$  to guess whether  $\text{decaps}(ct, sk) = \perp$ . If  $guess = 0$ , we return  $\perp$  in  $\mathcal{O}^{\text{Dec}}$ . Otherwise, for a valid ciphertext  $\bar{c}$  such that  $\perp \neq K \leftarrow \text{decaps}(sk, \bar{c})$ , the Dec oracle should return  $H_D(H_1(\overline{HS}, \bar{t}), H_D(H_2(\overline{HS}, \bar{t})))$ , where  $\overline{HS} = G(K)$  and  $\bar{t} = H_T(\bar{ct}, \bar{n})$ . As discussed in the ROM, we directly reprogram  $H_{12}(\overline{HS}, \bar{t})$  to random values  $\Theta = (\overline{chts}, \overline{shts})$ , and then output  $H_D(\overline{chts}), H_D(\overline{shts})$  in the  $\mathcal{O}^{\text{Dec}}$  oracle. Intuitively, the simulation is perfect if we reprogram  $H_{12}(\overline{HS}, \bar{t})$  to  $\Theta$  when the adversary  $\mathcal{A}$  first queries  $(\overline{HS}, H_T(\bar{ct}, \bar{n}))$  to either  $H_1$  or  $H_2$ . In the ROM, the probability of a correct random guess is  $1/q_{H_{12}}$ . However, in the QROM, since the adversary's queries are superposition RO-queries, it is difficult to define when the adversary makes a query  $H_{12}(\overline{HS}, \bar{t})$  to either  $H_1$  or  $H_2$ . Therefore, in the QROM, we adopt the approach from [21] to argue it differently. We observe that the consistency between  $\mathcal{O}^{\text{Dec}}$  and  $H_{12}$  can be ensured if the predicate  $\mathcal{O}^{\text{Dec}}(\bar{ct}, \bar{n}) = (H_D(H_1(\overline{HS}, \bar{t}), H_D(H_2(\overline{HS}, \bar{t}))))$  holds true. In the practical implementation of the decryption oracle  $\mathcal{O}^{\text{Dec}}(\bar{ct}, \bar{n})$ , there is an implicit classical query to  $H_{12}$ , which is omitted during the oracle's simulation in  $\mathcal{O}^{\text{Dec}}$ . Therefore, we utilize the refined optional query measure-and-reprogram technique [21] to argue this simulation impact.

Now we consider how to simulate the  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\bar{ct}, \bar{n}, \text{txt}, \text{tag})$  oracle without the secret key. Note that  $(\bar{ct}, \bar{n}, \cdot, \cdot)$  remains consistent with the  $\mathcal{O}^{\text{Dec}}(\bar{ct}, \bar{n})$  oracle, as defined in the IND-1CCA-MAC\* game. If the  $guess$  value in  $\mathcal{O}^{\text{Dec}}$  is 0, we correspondingly return  $\perp$  in  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$ . Otherwise, for a valid ciphertext  $\bar{c}$ , in the ROM, if  $\text{MAC.Vrf}(fk_S, \text{txt}, \text{tag})$  is correct, where  $fk_S = H_D(\overline{shts})$ , we can assert that the adversary has already queried the corresponding  $H_2$  with high probability. Subsequently, using the guess  $i$  from the simulation of  $\mathcal{O}^{\text{Dec}}$ , we extract the  $i + 1$ -th query directly and output the respective  $HS_{i+1}$ .

In the QROM, we simulate  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  by combining the refined optional-query measure-and-reprogram technique and OW2H technique. During the simulation of  $\mathcal{O}^{\text{Dec}}(\bar{ct}, \bar{n})$ , we have utilized the refined optional-query measure-and-

reprogram technique to  $H_{12}$ . Specifically, we measure the input quantum state to obtain  $x$  and subsequently reprogramming  $H_{12}$  at  $x$  to  $\Theta$  conditioned on some random values. If  $\text{MAC.Vrf}(fk_S, \text{txt}, \text{tag})$  is correct, we need to return HS in  $\mathcal{O}^{\text{Dec}}$ , where  $fk_S = H_D(\overline{sh\text{ts}})$ . Intuitively, we can use  $x$  (which includes HS) to directly output HS. However, it is important to discuss the order of  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  query and the measurement in the optional-query measure-and-reprogram technique. If  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  query occurs after the measurement, we can easily use  $x$  to perfectly simulate the output of HS. However, if  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  query occurs before the measurement, we cannot return HS since we have not yet measured it to obtain  $x$ . Nonetheless, this scenario implies that all  $H_{12}$  queries made by the adversary have not been reprogrammed and, consequently, are unrelated to the reprogrammed values  $\Theta$ , which is utilized to derive the MAC key. Therefore, we can argue that the adversary  $\mathcal{A}$  cannot distinguish the MAC key  $fk_S$  from a random value using the OW2H Lemma. Consequently, the adversary cannot forge a valid tag based on the security of MAC. Therefore, if  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  query occurs before the measurement, we directly output  $\perp$  in  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$ .

## 2 Preliminaries

### 2.1 Notation

The security parameter is denoted as  $\lambda$ . The set  $\{0, \dots, q\}$  is denoted as  $[q]$ . PPT is denoted to represent probabilistic polynomial time.  $\mathcal{K}$ ,  $\mathcal{M}$  and  $\mathcal{C}$  denote the key space, message space, and ciphertext space, respectively. For a finite set  $X$ ,  $x \leftarrow X$  represents the sampling of a uniformly random element from  $X$ .  $\Pr[P : G]$  indicates the probability that the predicate holds true when free variables in  $P$  are assigned according to the program in  $G$ . The sampling from some distribution  $D$  is represented by  $x \leftarrow D$ . For a quantum or randomized classical algorithm (resp. deterministic)  $A$ ,  $y \leftarrow A(x)$  (resp.  $y \leftarrow A(x)$ ) denotes that  $A$  outputs  $y$  on input  $x$ .  $x =?y$  is denoted as an integer that is 1 if  $x = y$ , and 0 otherwise.  $|X|$  denotes the cardinality of set  $X$ .  $A^H$  (resp.  $A^{|H}$ ) denotes that algorithm  $A$  gains classical (resp. quantum) access to the oracle  $H$ .

### 2.2 Cryptographic Primitives

**Definition 1 (Deterministic Public-Key Encryption).** *A DPKE over  $\mathcal{M}$  is a tuple of three algorithms  $\text{gen}, \text{enc}, \text{dec}$ . (1)  $(\text{pk}, \text{sk}) \leftarrow \text{gen}(1^\lambda)$ : The key generation algorithm  $\text{gen}$  takes as inputs the security parameter and outputs a key pair  $(\text{pk}, \text{sk})$ . Usually, we will omit the input of  $\text{gen}$  for brevity. (2)  $\text{ct} \leftarrow \text{enc}(\text{pk}, m)$ : The encryption algorithm takes as inputs the public key  $\text{pk}$  and a message  $m \in \mathcal{M}$  and deterministically outputs a ciphertext  $\text{ct}$ . (3)  $m' \leftarrow \text{dec}(\text{sk}, \text{ct})$ : The decryption algorithm, on input the secret key  $\text{sk}$  and the ciphertext  $\text{ct}$ , deterministically outputs a message  $m' \in \mathcal{M} \cup \{\perp\}$ .*

*Correctness.* A Deterministic Public-Key Encryption (DPKE) is  $\delta$ -correct if  $E[\max_{m \in M} \Pr[\text{dec}(\text{sk}, \text{ct}) \neq m : \text{ct} \leftarrow \text{enc}(\text{pk}, m)]] \leq \delta$ , where the expectation is taken over  $(\text{pk}, \text{sk}) \leftarrow \text{gen}$ . We say a DPKE is perfectly correct if  $\delta = 0$ .

*Rigidity.* [6]<sup>3</sup> A DPKE is *rigid* if for all key pairs  $(\text{pk}, \text{sk}) \leftarrow \text{gen}$ , and all ciphertexts  $\text{ct}$ , it holds that either  $\text{dec}(\text{sk}, \text{ct}) = \perp$  or  $\text{enc}(\text{pk}, \text{dec}(\text{sk}, \text{ct})) = \text{ct}$ .

**Definition 2 (OW-CPA-secure DPKE).** A DPKE scheme  $\text{DPKE} = (\text{gen}, \text{enc}, \text{dec})$  is OW-CPA if for any PPT adversary  $\mathcal{A}$  we have

$$\begin{aligned} \text{Adv}_{\text{DPKE}}^{\text{OW-CPA}}(\mathcal{A}) &= \Pr \left[ \mathcal{A}(\text{pk}, \text{ct}^*) \Rightarrow m^* : \begin{array}{l} (\text{pk}, \text{sk}) \leftarrow \text{gen}; \\ m^* \leftarrow \mathcal{M}; \text{ct}^* = \text{enc}(\text{pk}, m^*) \end{array} \right] \\ &= \text{negl}(\lambda), \end{aligned}$$

where the probability is taken over the randomness of the public-key generation and the adversary  $\mathcal{A}$ .

**Definition 3 (Key Encapsulation Mechanism).** A KEM over  $\mathcal{K}$  is a tuple of three algorithms  $\text{gen}, \text{encaps}, \text{decaps}$ . (1)  $(\text{pk}, \text{sk}) \leftarrow \text{gen}(1^\lambda)$ : The key generation algorithm  $\text{gen}$  takes as inputs the security parameter and outputs a key pair  $(\text{pk}, \text{sk})$ . Usually, we will omit the input of  $\text{gen}$  for brevity. (2)  $(\text{ct}, K) \leftarrow \text{encaps}(\text{pk})$ : The encapsulation algorithm takes as inputs the public key  $\text{pk}$  and it outputs a tuple  $(\text{ct}, K)$ , where  $K \in \mathcal{K}$  and  $\text{ct} \in \mathcal{C}$ . (3)  $K' \leftarrow \text{decaps}(\text{sk}, \text{ct})$ : The decapsulation procedure, on input the secret key  $\text{sk}$  and the ciphertext  $\text{ct}$ , outputs a key  $K'$ . If the KEM allows explicit rejection, the output is a key  $K' \in \mathcal{K}$  or the rejection symbol  $\perp$ .

**Definition 4 (OW-CPA-secure KEM).** A KEM scheme  $\text{KEM} = (\text{gen}, \text{encaps}, \text{decaps})$  is OW-CPA if for any PPT adversary  $\mathcal{A}$  we have

$$\begin{aligned} \text{Adv}_{\text{KEM}}^{\text{OW-CPA}}(\mathcal{A}) &= \Pr \left[ \mathcal{A}(\text{pk}, \text{ct}^*) \Rightarrow K^* : \begin{array}{l} (\text{pk}, \text{sk}) \leftarrow \text{gen}; \\ (K^*, \text{ct}^*) \leftarrow \text{encaps}(\text{pk}) \end{array} \right] \\ &= \text{negl}(\lambda), \end{aligned}$$

where the probability is taken over the randomness of the public-key generation, encapsulation, and the adversary  $\mathcal{A}$ .

It is straightforward to construct an OW-CPA-secure  $\text{KEM} = (\text{gen}, \text{encaps}, \text{decaps})$  based on an OW-CPA-secure  $\text{DPKE} = (\text{gen}', \text{enc}', \text{dec}')$  using the simple CPA transform shown in Fig. 1. In this paper, we denote this particular construction of KEM as DKEM or D-OW-CPA KEM.

---

<sup>3</sup> The NIST-PQC Round-3 Finalist NTRU [11] and NIST-PQC Round-4 Candidate Classic McEliece [2], are based on rigid one-way secure deterministic PKEs. For a general deterministic PKE, the rigid property can be achieved through a re-encryption transform.



**Definition 5 (IND-CPA-secure KEM).** A KEM scheme  $\text{KEM} = (\text{gen}, \text{encaps}, \text{decaps})$  is IND-CPA if for any PPT adversary  $\mathcal{A}$  we have

$$\text{Adv}_{\text{KEM}}^{\text{IND-CPA}}(\mathcal{A}) = \Pr \left[ \mathcal{A}(\text{pk}, \text{ct}^*, K_b^*) \Rightarrow b : \begin{array}{l} (\text{pk}, \text{sk}) \leftarrow_{\$} \text{gen}; b \leftarrow_{\$} \{0, 1\}; \\ (K_0^*, \text{ct}^*) \leftarrow_{\$} \text{encaps}(\text{pk}); K_1^* \leftarrow_{\$} \mathcal{K} \end{array} \right] = \text{negl}(\lambda),$$

**Definition 6 (MAC EUF-0T).** Let  $\text{MAC} = (\text{MAC.Vrf}, \text{MAC.Tag})$  be a message authentication code scheme (MAC). We say MAC is EUF-0T if for any PPT adversary  $\mathcal{A}$ ,  $\text{Adv}_{\text{EUF-0T}}^{\text{MAC}}(\mathcal{A}) := \Pr[\text{MAC.Vrf}(K, m, \text{tag}) = 1 : (m, \text{tag}) \leftarrow \mathcal{A}; K \leftarrow \mathcal{K}]$  is negligible in the security parameter, where the probability is taken over the sampling of the key and the randomness of the adversary.

**Definition 7 (IND-1CCA-MAC [20]).** We consider the games defined in Fig. 3. Let  $\mathcal{K}$  be the key space,  $G, H_1, H_2, H_3, H_4,$  and  $H_D$  be key-derivation functions with images in  $\{0, 1\}^n$ ,  $H_T$  be a hash function with images in  $\{0, 1\}^n$ , and a MAC scheme MAC. A KEM scheme  $\text{KEM} = (\text{gen}, \text{encaps}, \text{decaps})$  is IND-1CCA-MAC if for any PPT adversary  $\mathcal{A}$  we have

$$\text{Adv}_{\text{KEM}}^{\text{IND-1CCA-MAC}}(\mathcal{A}) := \left| \Pr[\text{IND-1CCA-MAC}_{\text{KEM}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right| = \text{negl}(\lambda)$$

where  $\Pr[\text{IND-1CCA-MAC}_{\text{KEM}}^b(\mathcal{A}) \Rightarrow 1]$  is the probability that  $\mathcal{A}$  wins the IND-1CCA-MAC $_b^{\text{KEM}}(\mathcal{A})$  game defined in Fig. 3.

In this game, the adversary receives a public key, a challenge ciphertext  $\text{ct}^*$  encapsulating a key  $K^*$ , a nonce  $n^*$ , and access two oracles,  $\mathcal{O}^{\text{Dec}}$  and  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$ , which it can query at most once to distinguish between three secrets:  $(\text{CHTS}_0, \text{SHTS}_0, \text{dHS}_0)$  derived from  $K^*, \text{ct}^*$  and  $n^*$ , or three random secrets  $(\text{CHTS}_1, \text{SHTS}_1, \text{dHS}_1)$ . It is important to note that these three secrets  $(\text{CHTS}_0, \text{SHTS}_0, \text{dHS}_0)$  are computed in a manner nearly identical to that of their similarly named counterparts in the modified TLS 1.3 protocol (see Fig. 4). The  $\mathcal{O}^{\text{Dec}}$  oracle takes a ciphertext (different from the challenge ciphertext) and serves as a decapsulation oracle, implementing a key schedule similar to that used in TLS to process the decapsulated key. Ultimately,  $\mathcal{O}^{\text{Dec}}$  returns two secrets:  $tk_C$  and  $tk_S$ . The  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  oracle takes a ciphertext (different from the challenge ciphertext), a tag, and some message  $\text{txt}$ . The ciphertext is then decrypted to recover a secret  $\text{HS}'$ , which is then subjected to a key schedule to generate a MAC key  $fk_S$ . Finally, the oracle verifies whether the tag constitutes a valid MAC on the  $\text{txt}$  with the key  $fk_S$ . If the tag is valid, it returns  $\text{HS}'$ . Otherwise, it returns an error  $\perp$ .

**Definition 8 (IND-1CCA-MAC\*).** The security definition is exactly the same as IND-1CCA-MAC, except that it restricts the adversary  $\mathcal{A}$  to make only the first query to  $\mathcal{O}^{\text{Dec}}$ , and subsequent query to  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  must have inputs  $(\text{ct}, n, \text{tag}, \text{txt})$  consistent with those in  $\mathcal{O}^{\text{Dec}}$ . Specifically, the input components  $(\text{ct}, n)$  for both  $\mathcal{O}^{\text{Dec}}$  and  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  must be the same.

---

IND-1CCA-MAC<sub>KEM</sub>( $\mathcal{A}$ )

```

 $b \leftarrow \{0, 1\}$ 
 $(pk, sk) \leftarrow \text{gen}$ 
 $(ct^*, K^*) \leftarrow \text{encaps}(pk)$ 
 $n^* \leftarrow \{0, 1\}^n$ 
 $HS^* \leftarrow G(K^*)$ 
 $CHTS_0 \leftarrow H_1(HS^*, H_T(ct^*, n^*))$ 
 $SHTS_0 \leftarrow H_2(HS^*, H_T(ct^*, n^*))$ 
 $dHS_0 \leftarrow H_3(HS^*)$ 
 $(CHTS_1, SHTS_1, dHS_1) \leftarrow \{0, 1\}^{3n}$ 
 $b' \leftarrow \mathcal{A}^{\text{O}^{\text{Dec}}, \text{O}^{\text{MAC}}}(pk, ct^*, n^*, (CHTS_b, SHTS_b, dHS_b))$ 
return  $1_{b'=b}$ 

```

---

Oracle  $\text{O}^{\text{Dec}}(ct, n)$ 

```

if more than 1 query : return  $\perp$ 
if  $(ct, n) = (ct^*, n^*)$  : return  $\perp$ 
 $K' \leftarrow \text{decaps}(sk, ct)$ 
if  $K' = \perp$  :
  return  $\perp$ 
 $HS' \leftarrow G(K')$ 
 $CHTS \leftarrow H_1(HS', H_T(ct, n))$ 
 $SHTS \leftarrow H_2(HS', H_T(ct, n))$ 
 $tk_C \leftarrow H_D(CHTS)$ ;  $tk_S \leftarrow H_D(SHTS)$ 
return  $(tk_C, tk_S)$ 

```

---

Oracle  $\text{O}^{\text{Dec}}_{\text{MAC}}(ct, n, \text{tag}, \text{txt})$ 

```

if more than 1 query : return  $\perp$ 
if  $(ct, n) = (ct^*, n^*)$  : return  $\perp$ 
 $K' \leftarrow \text{decaps}(sk, ct)$ 
 $HS' \leftarrow G(K')$ 
 $SHTS \leftarrow H_2(HS', H_T(ct, n))$ 
 $fk_S \leftarrow H_4(SHTS)$ 
if  $\text{MAC.Vrf}(fk_S, \text{txt}, \text{tag}) = \text{true}$  :
  return  $HS'$ 
return  $\perp$ 

```

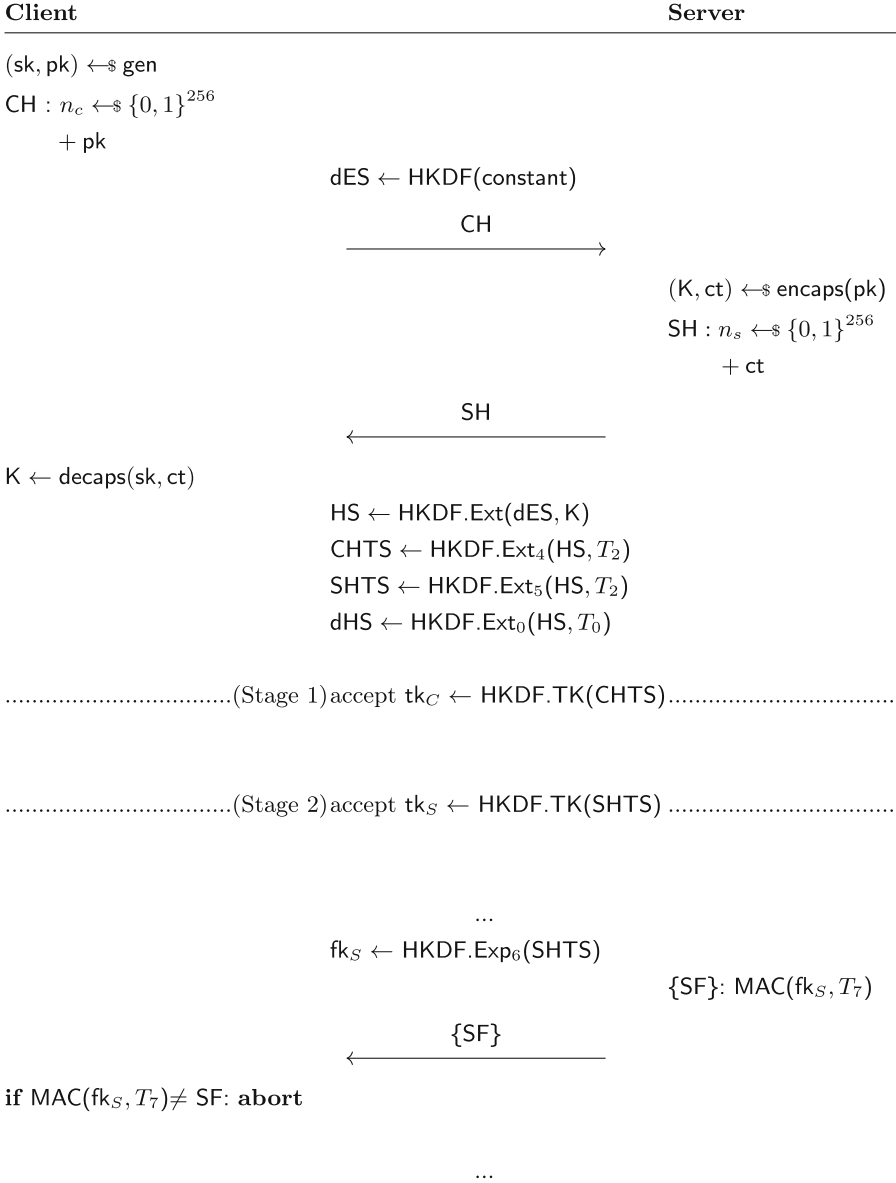
**Fig. 3.** IND-1CCA-MAC game

### 2.3 TLS1.3 Protocol

The Transport Layer Security (TLS) protocol is one of the most widely deployed cryptographic protocols in practice. In this paper, we focus on the TLS 1.3 handshake protocol. We refer the reader to [15] for a detailed introduction to TLS 1.3 handshake protocol. Additionally, we recall the notion of the MultiStage security model [15, 20] in complete version of the paper. We present the (full 1-RTT) handshake of TLS 1.3 with the DH component substituted by a KEM, as shown in Fig. 4. Below, we explicitly demonstrate the correspondence between TLS 1.3 and the IND-1CCA-MAC/IND-1CCA-MAC\* game. First, based on  $T_2 = H(\text{CH}, \text{SH}) = H(n_c, pk, n_s, ct)$ , we can rewrite  $\text{CHTS} = \text{HKDF.Ext}_4(\text{HS}, T_2)$  as  $\text{CHTS} = H_j(\text{HS}, H_T(n_s, ct))$ ,  $j \in \{1, 2\}$ , where  $H_j$  and  $H_T$  are both modeled as random oracles. Here, we omit the public key  $pk$  and the client nonce  $n_c$  because these values are not important for the proof of Theorem 6, game  $G_{B.2}$ . Similarly, since  $\text{dES}$  is constant, one can represent  $\text{HKDF.Ext}(\text{dES}, \cdot)$  as  $G(\cdot)$ ,  $\text{HKDF.Ext}_0(T_0, \cdot)$  as  $H_3(\cdot)$ ,  $\text{HKDF.Exp}_6(\cdot)$  as  $H_4(\cdot)$ , and  $\text{HKDF.TK}(\cdot)$  as  $H_D(\cdot)$ , where  $G$ ,  $H_3$ ,  $H_4$ , and  $H_D$  are all modeled as random oracles. This rewrite clarifies how these key steps in TLS 1.3 correspond precisely with the IND-1CCA-MAC/IND-1CCA-MAC\* game.

## 3 CPA-Secure KEMs Are Sufficient for TLS 1.3 in the ROM/QROM

In this chapter, we first prove OW-CPA/IND-CPA/D-OW-CPA KEMs are IND-1CCA-MAC\* with an EUF-0T secure MAC in the ROM and QROM. We



**Fig. 4.** TLS 1.3 handshake with KEM [20].  $\{\dots\}$  denotes a message encrypted with the session traffic key  $tk_S$ .  $T_i$  denotes the hash of the transcript up to the  $i$ -th message. For simplicity, the CH (resp. SH) message captures both the ClientHello and ClientKeyShare (resp. ServerHello and ServerKeyShare). Only the relevant steps for the proof are depicted. Keys in the remaining stages (3–6, not shown) are all derived from the Diffie-Hellman secret (dHS).

then establish the security proof of TLS 1.3 from IND-1CCA-MAC\* in the standard model. By integrating these results, we establish a comprehensive security proof for TLS 1.3 from OW-CPA/IND-CPA/D-OW-CPA KEMs in the ROM and QROM. Our analysis yields significantly tighter bounds than those presented in [20] in the ROM and is the first security proof for TLS 1.3 from CPA-secure KEMs in the QROM.

### 3.1 OW-CPA/IND-CPA/D-OW-CPA KEMs Imply IND-1CCA-MAC/IND-1CCA-MAC\* in the ROM

In this section, we demonstrate that OW-CPA/IND-CPA KEMs are IND-1CCA-MAC/IND-1CCA-MAC\* secure with an EUF-0T secure MAC in the ROM and D-OW-CPA KEMs are also IND-1CCA-MAC/IND-1CCA-MAC\* secure with an EUF-0T secure MAC in the ROM with tight reduction. More precisely, the KDFs  $G$ ,  $H_1$ ,  $H_2$ ,  $H_3$ ,  $H_4$ , and  $H_D$ , and the hash function  $H_T$  in the IND-1CCA-MAC/IND-1CCA-MAC\* games are assumed to be random oracles.

**Theorem 1.** *Let  $\text{KEM} = (\text{gen}, \text{encaps}, \text{decaps})$  be a KEM. Let the KDFs and the hash function in the IND-1CCA-MAC game be modeled as random oracles. Then, for any PPT adversary  $\mathcal{A}$  making at most  $q_G$ ,  $q_{H_1}$ ,  $q_{H_2}$ ,  $q_{H_3}$ ,  $q_{H_4}$ ,  $q_{H_D}$ ,  $q_{H_T}$  queries to  $G$ ,  $H_1$ ,  $H_2$ ,  $H_3$ ,  $H_4$ ,  $H_D$ ,  $H_T$  respectively, there exists an OW-CPA adversary  $\mathcal{C}$ , an IND-CPA adversary  $\mathcal{D}$ , and an EUF-0T adversary  $\mathcal{B}$  such that*

$$\begin{aligned} \text{Adv}_{\text{KEM}}^{\text{IND-1CCA-MAC}}(\mathcal{A}) &\leq 2q_G(q_{H_2} + 1)(q_{H_1} + q_{H_2} + 1) \cdot \text{Adv}_{\text{KEM}}^{\text{OW-CPA}}(\mathcal{C}) \\ &\quad + \text{Adv}_{\text{MAC}}^{\text{EUF-0T}}(\mathcal{B}) + \frac{q_{H_1} + 2q_{H_2} + q_{H_3} + q_{H_D} + q_{H_T} + 6}{2^n} \end{aligned}$$

$$\begin{aligned} \text{Adv}_{\text{KEM}}^{\text{IND-1CCA-MAC}}(\mathcal{A}) &\leq 4(q_{H_2} + 1)(q_{H_1} + q_{H_2} + 1) \cdot \text{Adv}_{\text{KEM}}^{\text{IND-CPA}}(\mathcal{D}) \\ &\quad + \text{Adv}_{\text{MAC}}^{\text{EUF-0T}}(\mathcal{B}) + \frac{q_{H_1} + 2q_{H_2} + q_{H_3} + q_{H_D} + q_{H_T} + 6}{2^n} \\ &\quad + \frac{4q_G(q_{H_2} + 1)(q_{H_1} + q_{H_2} + 1)}{|K|}. \end{aligned}$$

where  $\mathcal{B}$ ,  $\mathcal{C}$ , and  $\mathcal{D}$  have approximately the same running time as  $\mathcal{A}$ .

*Proof.* Let  $\mathcal{A}$  be an adversary against the IND-1CCA-MAC security of KEM, issuing (exactly) one classical query to  $\text{O}_{\text{MAC}}^{\text{Dec}}$  and one classical query to  $\text{O}^{\text{Dec}}$  (by introducing a dummy query if necessary). We proceed with a sequence of games, which are given in detail in Fig. 5, 6.

$\text{GAME}_{G_0}$ . This is the original IND-1CCA-MAC game. From now on, we assume w.l.o.g. that each query to ROs is unique (i.e., they never repeat). Thus,  $|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - 1/2| = \text{Adv}_{\text{KEM}}^{\text{IND-1CCA-MAC}}(\mathcal{A})$ .

GAMES $G_0 - G_8$	
1 :	$b \leftarrow_{\$} \{0, 1\}, (\text{pk}, \text{sk}) \leftarrow \text{gen}, G, H_{k \in \{1-4, T, D\}} \leftarrow_{\$} \Omega_G, \Omega_{H_i}, H_{12} \leftarrow_{\$} \Omega_{H_{12}}$
2 :	$(\text{ct}^*, K^*) \leftarrow_{\$} \text{encaps}(\text{pk}), (n^*, \overline{\text{chts}}, \overline{\text{shts}}) \leftarrow_{\$} \{0, 1\}^{3n}$
3 :	$\text{guess} \leftarrow_{\$} \{0, 1\}, i \leftarrow_{\$} [q_{H_1} + q_{H_2}], j \leftarrow_{\$} [q_{H_2}]$
4 :	$\text{HS}^* \leftarrow G(K^*), \text{dHS}_0 \leftarrow H_3(\text{HS}^*)$
5 :	$\text{CHTS}_0 \leftarrow H_1(\text{HS}^*, H_T(\text{ct}^*, n^*)), \text{SHTS}_0 \leftarrow H_2(\text{HS}^*, H_T(\text{ct}^*, n^*))$
6 :	$(\text{CHTS}_0, \text{SHTS}_0, \text{dHS}_0) \leftarrow_{\$} \{0, 1\}^{3n} \quad //G_6-$
7 :	$(\text{CHTS}_1, \text{SHTS}_1, \text{dHS}_1) \leftarrow_{\$} \{0, 1\}^{3n}$
8 :	$b' \leftarrow \mathcal{A}^{G, H_i, \mathcal{O}^{\text{Dec}}, \mathcal{O}_{\text{MAC}}^{\text{Dec}}}(\text{pk}, \text{ct}^*, n^*, (\text{CHTS}_b, \text{SHTS}_b, \text{dHS}_b))$
9 :	<b>if</b> $\exists (\text{ct}, n) \neq (\text{ct}^*, n^*), H_T(\text{ct}, n) = H_T(\text{ct}^*, n^*)$ : <b>abort</b> $//G_1-$
10 :	<b>if</b> $\mathcal{A}$ queries $H_{k_1}(\text{HS}^*, H_T(\text{ct}^*, n^*)), k_1 \in \{1, 2\}$ or $H_3(\text{HS}^*)$ : $//\text{QUERY}$
11 :	<b>if</b> $\mathcal{A}$ did not query $G(K^*)$ : <b>abort</b> $//G_5-$
12 :	<b>return</b> $1_{b=b'}$
$H_{k_1}(\text{HS}, t) \quad //G_4-$	$H_{12}(\text{HS}, t) \quad //G_7-$
1 :	$(\text{CHTS}, \text{SHTS}) = H_{12}(\text{HS}, t)$
2 :	<b>if</b> $k_1 = 1$ <b>return</b> $\text{CHTS}$
3 :	<b>else return</b> $\text{SHTS}$
1 :	$//$ define $\text{HS}_{l+1} = \text{HS}$
2 :	<b>if</b> $l = i$ <b>return</b> $\overline{\text{chts}}, \overline{\text{shts}}$
3 :	$l = l + 1$
4 :	<b>return</b> $H_{12}(\text{HS}, t)$

**Fig. 5.** Games for the proof of Theorem 1

GAME $G_1$ : In game  $G_1$ , we abort if there exists  $(\text{ct}, n) \neq (\text{ct}^*, n^*)$  such that  $H_T(\text{ct}, n) = H_T(\text{ct}^*, n^*)$ . Since there are at most  $q_{H_T} + 4$  queries to  $H_T$  in the game (including two additional implicit queries to  $H_T$  in both  $\mathcal{O}^{\text{Dec}}$  and  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$ ), the probability of  $\mathcal{A}$  finding such  $(\text{ct}, n)$  given by  $(\text{ct}^*, n^*)$  is less than  $(q_{H_T} + 4)/2^n$ .

$$|\Pr[G_0^{\mathcal{A}} \Rightarrow 1] - \Pr[G_1^{\mathcal{A}} \Rightarrow 1]| \leq \frac{q_{H_T} + 4}{2^n}.$$

GAME $G_2$ . In game  $G_2$ , we abort whenever the MAC verification succeeds on the query  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$  but  $fk_S := H_4(\text{SHTS})$  was never queried before the query of  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$ , where  $\text{SHTS} := H_2(G(K), H_T(\text{ct}_2, n_2))$  and  $K := \text{Decaps}(\text{sk}, \text{ct}_2)$ . If this is the case, it means the MAC key  $fk_S := H_4(\text{SHTS})$  is indistinguishable from a random value for  $\mathcal{A}$  when it queries the  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$  oracle, but it manages to forge a valid tag. Therefore, we can build an adversary  $\mathcal{B}$  that breaks MAC unforgeability. More formally,  $\mathcal{B}$  samples a pair of keys  $(\text{sk}, \text{pk}) \leftarrow \text{gen}$  for KEM, generates a valid input for  $\mathcal{A}$  and simulates the  $\mathcal{O}^{\text{Dec}}$  oracle with the secret key. Then, when  $\mathcal{A}$  query  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$ ,  $\mathcal{B}$  outputs  $(\text{txt}, \text{tag})$  as a forgery.

We also abort if the value SHTS computed in the  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  oracle is such that  $\text{SHTS} = \text{SHTS}_b$ . Since  $G_1$  and  $G_2$  return  $\perp$  when  $H_T(\text{ct}_2, n_2) = H_T(\text{ct}^*, n^*)$ , the event  $\text{SHTS} = \text{SHTS}_b = H_2(\cdot, H_T(\text{ct}^*, n^*))$  occurs implies the adversary that given  $\text{SHTS}_b$  finds  $H_2(\cdot, H_T(\text{ct}_2, n_2)) = \text{SHTS}_b$  such that  $H_T(\text{ct}_2, n_2) \neq H_T(\text{ct}^*, n^*)$ . Note that there are at most  $q_{H_2} + 1$  queries to  $H_2$  (including an implicit query in  $\mathcal{O}^{\text{Dec}}$ ), the probability of the event  $\text{SHTS} = \text{SHTS}_b$  occurring is at most  $(q_{H_2} + 1)/2^n$ . Therefore, we have

$$|\Pr[G_1^{\mathcal{A}} \Rightarrow 1] - \Pr[G_2^{\mathcal{A}} \Rightarrow 1]| \leq \text{Adv}_{\text{MAC}}^{\text{EUF-OT}}(\mathcal{B}) + \frac{q_{H_2} + 1}{2^n}.$$

$\mathcal{O}^{\text{Dec}}(\text{ct}_1, n_1)$	$\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$
1: <b>if</b> more than 1 query :	1: <b>if</b> more than 1 query : <b>return</b> $\perp$
2: <b>return</b> $\perp$	2: <b>if</b> $(\text{ct}_2, n_2) = (\text{ct}^*, n^*)$ : <b>return</b> $\perp$
3: <b>if</b> $(\text{ct}_1, n_1) = (\text{ct}^*, n^*)$ :	3: <b>if</b> $(\text{ct}_2, n_2) = (\text{ct}_1, n_1)$ :   // $G_7-$
4: <b>return</b> $\perp$	4: <b>if</b> $\text{guess} = 0$ : <b>return</b> $\perp$
5: <b>if</b> $\text{guess} = 0$ :   // $G_7-$	5: $fk_S \leftarrow H_4(\overline{\text{shts}})$
6: <b>return</b> $\perp$	6: <b>if</b> $\text{MAC.Vrf}(fk_S, \text{txt}, \text{tag}) = \text{true}$ :
7: <b>return</b> $H_D(\overline{\text{chts}}), H_D(\overline{\text{shts}})$	7: <b>return</b> $\text{HS}_{i+1}$
8:     // $G_7-$	8: <b>return</b> $\perp$
9: $K' \leftarrow \text{decaps}(\text{sk}, \text{ct}_1)$	9: <b>if</b> $(\text{ct}_2, n_2) \neq (\text{ct}_1, n_1)$ // $G_8-$
10: <b>if</b> $K' = \perp$ : <b>return</b> $\perp$	10: $\forall \mathcal{A}$ did not query $\mathcal{O}^{\text{Dec}}$ :
11: $\text{HS}' \leftarrow G(K')$	11: <b>if</b> $j = q_{H_2}$ : <b>return</b> $\perp$
12: $\text{CHTS} \leftarrow H_1(\text{HS}', H_T(\text{ct}_1, n_1))$	12: <b>else return</b> $\text{HS}_{j+1}$
13: $\text{SHTS} \leftarrow H_2(\text{HS}', H_T(\text{ct}_1, n_1))$	13: $K' \leftarrow \text{decaps}(\text{sk}, \text{ct}_2)$ , $\text{HS}' \leftarrow G(K')$
14: $tk_C \leftarrow H_D(\text{CHTS})$	14: $\text{SHTS} \leftarrow H_2(\text{HS}', H_T(\text{ct}_2, n_2))$
15: $tk_S \leftarrow H_D(\text{SHTS})$	15: $fk_S \leftarrow H_4(\text{SHTS})$
16: <b>return</b> $tk_C, tk_S$	16: <b>if</b> $\text{SHTS} = \text{SHTS}_b$ : <b>abort</b> // $G_2-$
	17: <b>if</b> $\text{MAC.Vrf}(fk_S, \text{txt}, \text{tag}) = \text{true}$ :
	18: <b>if</b> $\mathcal{A}$ did not query $H_4(\text{SHTS})$ :
	19: <b>abort</b> // $G_2-$
	20: <b>if</b> $\mathcal{A}$ did not query
	21: $H_2(\text{HS}', H_T(\text{ct}_2, n_2))$ :
	22: <b>abort</b> // $G_3-$
	23: <b>return</b> $\text{HS}'$
	24: <b>return</b> $\perp$

**Fig. 6.** Games for the proof of Theorem 1

GAME  $G_3$ . In game  $G_3$ , we abort whenever the MAC verification succeeds on the query  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$  but  $H_2(G(K), H_T(\text{ct}_2, n_2))$  was never queried before the query of  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$ , where  $K := \text{decaps}(\text{sk}, \text{ct}_2)$ . By the previous game, it means that the adversary queried  $\text{SHTS} := H_2(G(K), H_T(\text{ct}_2, n_2))$  to  $H_4$  without having queried  $H_2(G(K), H_T(\text{ct}_2, n_2))$  beforehand. Let us analyze what information  $\mathcal{A}$  has about  $\text{SHTS} \neq \text{SHTS}_b$  if it did not query  $H_2(G(K), H_T(\text{ct}_2, n_2))$ . Note that the only potential “leakage” is from  $\mathcal{O}^{\text{Dec}}$  that returns  $tk_S := H_D(\text{SHTS})$ , where  $H_D$  is a RO perfectly hiding  $\text{SHTS}$ . If  $G_3$  aborts, we can construct an adversary  $\mathcal{A}$  who is capable of recovering  $\text{SHTS}$  given the random oracle output  $H_D(\text{SHTS})$ . The best strategy for  $\mathcal{A}$  to recover  $\text{SHTS}$  is to query random values  $x \in \{0, 1\}^n$  to  $H_D$  until it finds  $x$  such that  $H_D(x) = tk_S$ , or randomly guess the value of  $\text{SHTS}$ . Thus, the advantage of  $\mathcal{A}$  is at most  $\frac{q_{H_D} + 1}{2^n}$ . Hence, we have

$$|\Pr[G_2^{\mathcal{A}} \Rightarrow 1] - \Pr[G_3^{\mathcal{A}} \Rightarrow 1]| \leq \frac{q_{H_D} + 1}{2^n}.$$

GAME  $G_4$ . In game  $G_4$ , we use a new random oracle  $H_{12}(\text{HS}, t)$  to simulate both  $H_1$  and  $H_2$ . Let  $H_{12}(\text{HS}, t) := (H_1(\text{HS}, t), H_2(\text{HS}, t))$ . Here,  $H_{12}$  is a random oracle maintained internally by the challenger. When adversary  $\mathcal{A}$  queries  $H_1(\text{HS}, t)$ , the challenger only needs to query  $H_{12}(\text{HS}, t)$  and then output the first component. Similarly, when adversary  $\mathcal{A}$  queries  $H_2(\text{HS}, t)$ , the challenger only needs to query  $H_{12}(\text{HS}, t)$  and then output the second component. Consequently, the total number of queries to  $H_{12}$  is at most  $q_{H_1} + q_{H_2}$ . Therefore, this  $G_4$  is consistent with  $G_3$  from  $\mathcal{A}$ 's view. In other words, we have

$$\Pr[G_3^{\mathcal{A}} \Rightarrow 1] = \Pr[G_4^{\mathcal{A}} \Rightarrow 1].$$

GAME  $G_5$ . In game  $G_5$ , we abort whenever the adversary did not query  $G(K^*)$  (which is equal to  $\text{HS}^*$ ) but queried  $H_1(\text{HS}^*, H_T(\text{ct}^*, n^*))$ ,  $H_2(\text{HS}^*, H_T(\text{ct}^*, n^*))$ , or  $H_3(\text{HS}^*)$ . Note that the  $\mathcal{O}^{\text{Dec}}$  and  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  never query  $H_1(\text{HS}^*, H_T(\text{ct}^*, n^*))$ ,  $H_2(\text{HS}^*, H_T(\text{ct}^*, n^*))$ , or  $H_3(\text{HS}^*)$  and the challenge values given to  $\mathcal{A}$  are either perfectly random or completely hide  $\text{HS}^*$ . Similarly, the  $\mathcal{O}^{\text{Dec}}$  oracle also completely hides  $\text{HS}^*$ . According to the definition of the previous game, the  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$  oracle returns  $\text{HS}^*$  if and only if the corresponding  $(\text{HS}^*, H_T(\text{ct}_2, n_2))$  was queried to  $H_2$  before, where  $\text{HS}^* = G(K')$  and  $K' = \text{decaps}(\text{sk}, \text{ct}_2)$ . That is, the adversary has queried  $H_2(\text{HS}^*, H_T(\text{ct}_2, n_2))$  before the  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  oracle, which means the adversary already knows  $\text{HS}^*$ . Therefore,  $\text{HS}^*$  is uniformly random and independent from  $\mathcal{A}$ 's view, and the probability that  $\mathcal{A}$  queries  $\text{HS}^*$  to  $H_1$ ,  $H_2$ , or  $H_3$  is upper bounded by  $\frac{q_{H_1} + q_{H_2} + q_{H_3}}{2^n}$ . Hence, we have

$$|\Pr[G_4^{\mathcal{A}} \Rightarrow 1] - \Pr[G_5^{\mathcal{A}} \Rightarrow 1]| \leq \frac{q_{H_1} + q_{H_2} + q_{H_3}}{2^n}.$$

GAME $G_6$ . In game  $G_6$ ,  $(\text{CHTS}_0, \text{SHTS}_0, \text{dHS}_0)$  is replaced by  $(\text{CHTS}_0, \text{SHTS}_0, \text{dHS}_0) \leftarrow_{\$} \{0, 1\}^{3n}$ . Define QUERY as the event that  $H_1(\text{HS}^*, H_T(\text{ct}^*, n^*))$ ,  $H_2(\text{HS}^*, H_T(\text{ct}^*, n^*))$ , or  $H_3(\text{HS}^*)$  is queried by the adversary. Then,  $G_6$  is identical to  $G_5$  in  $\mathcal{A}$ 's view unless the event QUERY happens. Thus, we have

$$|\Pr[G_5^{\mathcal{A}} \Rightarrow 1] - \Pr[G_6^{\mathcal{A}} \Rightarrow 1]| \leq \Pr[\text{QUERY} : G_6].$$

It is evident that in game  $G_6$ , the bit  $b$  is independent of adversary  $\mathcal{A}$ 's view. Therefore, we have

$$\Pr[G_6^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

GAME $G_7$ . In game  $G_7$ , the challenger can simulate the  $\mathcal{O}^{\text{Dec}}(\text{ct}_1, n_1)$  oracle without the secret key. Initially, we randomly sample  $guess \leftarrow_{\$} \{0, 1\}$  to guess whether  $K' = \text{decaps}(\text{sk}, \text{ct}_1)$  equals to  $\perp$ . If  $guess = 0$ , in this case, we guess  $K' = \perp$ , so we just return  $\perp$  in  $\mathcal{O}^{\text{Dec}}$ . Otherwise, if  $guess = 1$ , we will make two further changes in this game. First, we modify the Dec oracle and replace  $\text{CHTS} := H_1(\text{HS}', H_T(\text{ct}_1, n_1))$  and  $\text{SHTS} := H_2(\text{HS}', H_T(\text{ct}_1, n_1))$  with  $\text{CHTS} = \overline{\text{chts}}$  and  $\text{SHTS} = \overline{\text{shts}}$ , where  $\overline{\text{chts}}$  and  $\overline{\text{shts}}$  are randomly chosen from  $\{0, 1\}^n$  and  $\text{HS}' = G(K')$ . Second, we reprogram the random oracle  $H_{12}$  conditionally on a uniform  $i$  over  $[q_{H_1} + q_{H_2}]$ . In other words, on the  $i + 1$ -th query, we reprogram  $H_{12}$  to return  $(\overline{\text{chts}}, \overline{\text{shts}})$ , while keeping all other queries unchanged. Let  $(i^* + 1)$  denote the number of first queries to  $H_{12}$  with  $(\text{HS}', H_T(\text{ct}_1, n_1))$ , where  $i^* \in [q_{H_1} + q_{H_2} - 1]$ . We also denote  $i^* = q_{H_1} + q_{H_2}$  as the event that  $H_{12}$  makes no queries with  $(\text{HS}', H_T(\text{ct}_1, n_1))$ .

Moreover, if the  $(\text{ct}_2, n_2, \dots)$  input to  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  is equal to the  $(\text{ct}_1, n_1)$  input to  $\mathcal{O}^{\text{Dec}}$ , that is,  $(\text{ct}_1, n_1) = (\text{ct}_2, n_2)$ . We change  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  as follows: In the case of  $guess = 0$ , we return  $\perp$  in  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$ . Otherwise, we compute  $fk_S = H_4(\overline{\text{shts}})$  and verify if  $\text{MAC.Vrf}(fk_S, \text{txt}, \text{tag})$  is correct. If the verification returns true, we only need to extract  $\text{HS}_{i+1}$  from the  $i + 1$ -th query to  $H_{12}$ , and then output  $\text{HS}_{i+1}$ . According to the previous game, when  $\text{MAC.Vrf}(fk_S, \text{txt}, \text{tag})$  is correct, we can conclude that the adversary has already queried the corresponding  $H_2$ , which means the  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  oracle query must have occurred after the  $i + 1$ -th  $H_{12}$  query so that we can extract the corresponding  $\text{HS}_{i+1}$ . If the verification is not true, we output  $\perp$ .

Note that  $G_7$  has the same distribution as  $G_6$  in  $\mathcal{A}$ 's view when the event  $i^* = i$  occurs and the  $guess$  is correct. Thus, we have

$$\Pr[\text{QUERY} : G_6] \leq 2(q_{H_1} + q_{H_2} + 1) \Pr[\text{QUERY} : G_7].$$

GAME  $G_8$ . In game  $G_8$ , the challenger can simulate  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$  oracle without the secret key. We modify the  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  oracle as follows: If the  $(\text{ct}_2, n_2, \dots)$  input to  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  is not equal to the  $(\text{ct}_1, n_1)$  input to  $\mathcal{O}^{\text{Dec}}$  or  $\mathcal{A}$  did not query  $\mathcal{O}^{\text{Dec}}$  before. According to the definition of the previous game, this oracle returns something other than  $\perp$  if and only if the corresponding  $(\text{HS}', H_T(\text{ct}_2, n_2))$  was queried to  $H_2$  before, where  $\text{HS}' = G(K')$  and  $K' = \text{decaps}(\text{sk}, \text{ct}_2)$ . Therefore, one can randomly choose  $j$  over  $[q_{H_2}]$  and guess



whether  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  outputs  $\perp$  (if  $j = q_{H_2}$ ) or corresponding  $\text{HS}'$  is in the  $j + 1$ -th<sup>4</sup> query and return  $\text{HS}_{j+1}$ . Therefore, the simulation is such that when  $j = q_{H_2}$ , output  $\perp$ . Otherwise, we extract  $\text{HS}_{j+1}$  from the  $j + 1$ -th query to  $H_2$  (simulated by  $H_{12}$ ), and then output  $\text{HS}_{j+1}$ . Overall, the simulation works with probability  $\frac{1}{(q_{H_2}+1)}$ . Thus, we have

$$\Pr[\text{QUERY} : G_7] \leq (q_{H_2} + 1) \Pr[\text{QUERY} : G_8]$$

Note that if QUERY happens,  $K^*$  will be in the  $G$ -list of queries made by  $\mathcal{A}$ . Let  $(\text{pk}, \text{sk}) \leftarrow \text{\$ gen}$ ,  $(K^*, \text{ct}^*) \leftarrow \text{\$ encaps}(\text{pk})$ . Then, we construct an adversary  $\mathcal{C}'(\text{pk}, \text{ct}^*)$  samples  $n^*, \text{chts}, \text{shts}, \text{guess}, i, j$  as in  $G_8$  and  $\text{CHTS}^*, \text{SHTS}^*, \text{dHS}^* \leftarrow \text{\$ } \{0, 1\}^{3n}$ . Then  $\mathcal{C}$  picks five  $q_{H_k}$ -wise ( $k \in \{12, 3, 4, H, T\}$ ) independent functions and a  $q_G$ -wise independent functions (indistinguishable from a random function for a  $q_{H_k}(q_G)$ -query adversary according to [33]) and runs  $\mathcal{A}^{G, H_i, \mathcal{O}^{\text{Dec}}, \mathcal{O}_{\text{MAC}}^{\text{Dec}}}(\text{pk}, \text{ct}^*, n^*, \text{CHTS}^*, \text{SHTS}^*, \text{dHS}^*)$  as in game  $G_8$  and returns  $\mathcal{A}$ 's  $G$ -query list  $G$ -List.

Now, we can construct an adversary  $\mathcal{C}$  against the OW-CPA security of the underlying KEM.  $\mathcal{C}$  runs  $\mathcal{C}'$  and randomly selects one message in the  $G$ -List as a return. Then, we have  $\text{Adv}_{\text{KEM}}^{\text{OW-CPA}}(\mathcal{C}) \geq \frac{1}{q_G} \Pr[\text{Query} : G_8]$ . Therefore, we have

$$\begin{aligned} \text{Adv}_{\text{KEM}}^{\text{IND-1CCA-MAC}}(\mathcal{A}) &\leq 2q_G(q_{H_2} + 1)(q_{H_1} + q_{H_2} + 1) \cdot \text{Adv}_{\text{KEM}}^{\text{OW-CPA}}(\mathcal{C}) \\ &\quad + \text{Adv}_{\text{MAC}}^{\text{EUF-OT}}(\mathcal{B}) + \frac{q_{H_1} + 2q_{H_2} + q_{H_3} + q_{H_D} + q_{H_T} + 6}{2^n}. \end{aligned}$$

Right now, we consider the case of the IND-CPA KEM. Specifically, the IND-CPA challenger generates  $(\text{pk}, \text{sk}) \leftarrow \text{\$ gen}$ ,  $(\text{ct}^*, K^*) \leftarrow \text{\$ encaps}(\text{pk})$ , and  $b' \leftarrow \text{\$ } \{0, 1\}$ . When  $b' = 0$ , define  $K_{b'}^* = K^*$ , and when  $b' = 1$ , define  $K_{b'}^* \leftarrow K$ . Finally,  $\mathcal{D}$  needs to guess the value of  $b'$  after receiving  $(\text{pk}, K_{b'}^*, \text{ct}^*)$  from the challenger.

Then,  $\mathcal{D}$  runs  $\mathcal{C}'(\text{pk}, \text{ct}^*)$  to get  $\mathcal{A}$ 's  $G$ -List. Let BADG be the event that the  $G$ -List contains  $K_1^*$ . Since  $K_1^*$  is uniformly random and independent from  $\mathcal{A}$ 's view, the probability that adversary  $\mathcal{A}$  queries  $G(K_1^*)$  is at most  $\frac{q_G}{|K|}$ . For the remainder of the proof, we assume BADG did not happen. If QUERY happens, this means adversary  $\mathcal{A}$  queried the random oracle  $G$  on  $K_0^*$ . In this case, if  $\mathcal{D}$  obtains  $K_{b'}^*$  in the  $G$ -List, it directly outputs  $b = 0$ ; otherwise, it outputs  $b = 1$ . If QUERY does not happen,  $\mathcal{D}$  uniformly randomly guesses the value of  $b'$ , i.e., it outputs  $b \leftarrow \text{\$ } \{0, 1\}$ . Thus, we have

$$\begin{aligned} \text{Adv}_{\text{KEM}}^{\text{IND-CPA}}(\mathcal{D}) + \frac{q_G}{|K|} &\geq |\Pr[b' = b] - \frac{1}{2}| \\ &= |\Pr[\text{QUERY}:G_8] + \frac{1}{2} \Pr[\neg\text{QUERY}:G_8] - \frac{1}{2}| \\ &= \frac{1}{2} \Pr[\text{QUERY}:G_8]. \end{aligned}$$

<sup>4</sup> Here, we exclude the  $H_{12}$  query induced by the  $H_1$  query from the adversary.

Putting the bounds together, we have

$$\begin{aligned} \text{Adv}_{\text{KEM}}^{\text{IND-1CCA-MAC}}(\mathcal{A}) &\leq 4(q_{H_2} + 1)(q_{H_1} + q_{H_2} + 1) \cdot \text{Adv}_{\text{KEM}}^{\text{IND-CPA}}(\mathcal{D}) \\ &\quad + \text{Adv}_{\text{MAC}}^{\text{EUF-OT}}(\mathcal{B}) + \frac{q_{H_1} + 2q_{H_2} + q_{H_3} + q_{H_D} + q_{H_T} + 6}{2^n} \\ &\quad + \frac{4q_G(q_{H_2} + 1)(q_{H_1} + q_{H_2} + 1)}{|K|}. \end{aligned}$$

□

**Theorem 2.** *Let  $\text{KEM} = (\text{gen}, \text{encaps}, \text{decaps})$  be a KEM. Let the KDFs and the hash function in the  $\text{IND-1CCA-MAC}^*$  game be modeled as random oracles. Then, for any PPT adversary  $\mathcal{A}$  making at most  $q_G, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{H_D}, q_{H_T}$  queries to  $G, H_1, H_2, H_3, H_4, H_D, H_T$  respectively, there exists a OW-CPA adversary  $\mathcal{C}$ , an IND-CPA adversary  $\mathcal{D}$  and an EUF-OT adversary  $\mathcal{B}$  such that*

$$\begin{aligned} \text{Adv}_{\text{KEM}}^{\text{IND-1CCA-MAC}^*}(\mathcal{A}) &\leq 2q_G(q_{H_1} + q_{H_2} + 1) \cdot \text{Adv}_{\text{KEM}}^{\text{OW-CPA}}(\mathcal{C}) \\ &\quad + \text{Adv}_{\text{MAC}}^{\text{EUF-OT}}(\mathcal{B}) + \frac{q_{H_1} + 2q_{H_2} + q_{H_3} + q_{H_D} + q_{H_T} + 6}{2^n} \end{aligned}$$

$$\begin{aligned} \text{Adv}_{\text{KEM}}^{\text{IND-1CCA-MAC}^*}(\mathcal{A}) &\leq 4(q_{H_1} + q_{H_2} + 1) \cdot \text{Adv}_{\text{KEM}}^{\text{IND-CPA}}(\mathcal{D}) \\ &\quad + \text{Adv}_{\text{MAC}}^{\text{EUF-OT}}(\mathcal{B}) + \frac{q_{H_1} + 2q_{H_2} + q_{H_3} + q_{H_D} + q_{H_T} + 6}{2^n} \\ &\quad + \frac{4q_G(q_{H_2} + 1)(q_{H_1} + q_{H_2} + 1)}{|K|}. \end{aligned}$$

*Proof.* It is easy to see that, except for  $G_8$ , the proof is the same as the proof of Theorem 1. The  $G_8$  is redundant because  $\text{IND-1CCA-MAC}^*$  require  $(\text{ct}_1, n_1) = (\text{ct}_2, n_2)$  and the query  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  is subsequent to the query  $\mathcal{O}^{\text{Dec}}$ . Thus, in this proof, we can simply define  $G_8$  to be identical to  $G_7$  to maintain consistency with Theorem 1. Hence, we have  $\Pr[\text{QUERY} : G_7] = \Pr[\text{QUERY} : G_8]$ . □

**Theorem 3.** *Let  $\text{KEM} = (\text{gen}, \text{encaps}, \text{decaps}) = S(\text{DPKE}(\text{gen}', \text{enc}', \text{dec}'))^5$  be a DKEM and the underlying  $\delta$ -correctness DPKE is rigid. Let the KDFs and the hash function in the  $\text{IND-1CCA-MAC}$  game be modeled as random oracles. Then, for any PPT adversary  $\mathcal{A}$  making at most  $q_G, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{H_D}, q_{H_T}$  queries to  $G, H_1, H_2, H_3, H_4, H_D, H_T$  respectively, there exists an OW-CPA adversary  $\mathcal{C}$  and an EUF-OT adversary  $\mathcal{B}$  such that*

$$\begin{aligned} \text{Adv}_{\text{KEM}}^{\text{IND-1CCA-MAC}}(\mathcal{A}) &\leq 4\text{Adv}_{\text{DKEM}}^{\text{OW-CPA}}(\mathcal{C}) + \text{Adv}_{\text{MAC}}^{\text{EUF-OT}}(\mathcal{B}) \\ &\quad + \frac{q_{H_1} + 2q_{H_2} + q_{H_3} + q_{H_D} + q_{H_T} + 6}{2^n} + \delta \end{aligned}$$

where  $\mathcal{C}$  have approximately the same running time as  $\mathcal{A}$ .

GAMES $G_0 - G_9$
1 : $b \leftarrow \{0, 1\}, (\mathbf{pk}, \mathbf{sk}) \leftarrow \text{gen}, G, H_{i \in \{1-4, T, D\}} \leftarrow \Omega_G, \Omega_{H_i}$
2 : $n^* \leftarrow \{0, 1\}^n, \text{guess}, \text{guess}' \leftarrow \{0, 1\}$
3 : $(\text{ct}^*, K^*) \leftarrow \text{encaps}(\mathbf{pk}), \text{HS}^* \leftarrow G(K^*), \text{dHS}_0 \leftarrow H_3(\text{HS}^*)$
4 : $\text{CHTS}_0 \leftarrow H_1(\text{HS}^*, H_T(\text{ct}^*, n^*)), \text{SHTS}_0 \leftarrow H_2(\text{HS}^*, H_T(\text{ct}^*, n^*))$
5 : $(\text{CHTS}_0, \text{SHTS}_0, \text{dHS}_0) \leftarrow \{0, 1\}^{3n} \quad // G_5-$
6 : $(\text{CHTS}_1, \text{SHTS}_1, \text{dHS}_1) \leftarrow \{0, 1\}^{3n}$
7 : $b' \leftarrow \mathcal{A}^{G, H_i, \mathcal{O}^{\text{Dec}}, \mathcal{O}_{\text{MAC}}^{\text{Dec}}}(\mathbf{pk}, \text{ct}^*, n^*, (\text{CHTS}_b, \text{SHTS}_b, \text{dHS}_b))$
8 : <b>if</b> $\exists (\text{ct}, n) \neq (\text{ct}^*, n^*), H_T(\text{ct}, n) = H_T(\text{ct}^*, n^*)$ : <b>abort</b> $// G_1-$
9 : <b>if</b> $\mathcal{A}$ queries $H_k(\text{HS}^*, H_T(\text{ct}^*, n^*)), k \in \{1, 2\}$ or $H_3(\text{HS}^*)$ : $// \text{QUERY}$
10 : <b>if</b> $\mathcal{A}$ did not query $G(K^*)$ : <b>abort</b> $// G_4-$
11 : <b>if</b> $\exists (K, \text{HS}) \in \mathcal{L}_G$ s. th. $\text{dec}'(\mathbf{sk}, \text{enc}'(\mathbf{pk}, K)) \neq K$ : <b>abort</b> $// G_7-$
12 : <b>return</b> $1_{b=b'}$
$\mathcal{O}^{\text{Dec}}(\text{ct}_1, n_1)$
1 : <b>if</b> more than 1 query $\vee (\text{ct}_1, n_1) = (\text{ct}^*, n^*)$ : <b>return</b> $\perp$
2 : $K' \leftarrow \text{decaps}(\mathbf{sk}, \text{ct}_1) \quad // G_0 - G_7$
3 : <b>if</b> $K' = \perp$ : <b>return</b> $\perp \quad // G_0 - G_7$
4 : $\text{HS}' \leftarrow G(K') \quad // G_0 - G_7$
5 : <b>if</b> $\text{guess} = 0$ : <b>return</b> $\perp \quad // G_8-$
6 : <b>if</b> $\exists K'$ s. th. $(K', \text{ct}_1, \text{HS}') \in \mathcal{L}$ : <b>extract</b> $\text{HS}' \quad // G_8-$
7 : <b>else</b> $\text{HS}' \leftarrow \{0, 1\}^n, \mathcal{L}_{\text{Dec}} = \{\text{ct}_1, \text{HS}'\} \quad // G_8-$
8 : $\text{CHTS} \leftarrow H_1(\text{HS}', H_T(\text{ct}_1, n_1)), \text{SHTS} \leftarrow H_2(\text{HS}', H_T(\text{ct}_1, n_1))$
9 : $tk_C \leftarrow H_D(\text{CHTS}), tk_S \leftarrow H_D(\text{SHTS})$
10 : <b>return</b> $tk_C, tk_S$

**Fig. 7.** Games for the proof of Theorem 3

*Proof.* Let  $\mathcal{A}$  be an adversary against the IND-1CCA-MAC security of KEM, issuing one classical query to  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  and one classical query to  $\mathcal{O}^{\text{Dec}}$  (by introducing a dummy query if necessary). Let  $\text{DKEM} = S(\text{DPKE}(\text{gen}', \text{enc}', \text{dec}'))$ . In this proof, we utilize the rigid property to simulate the  $\mathcal{O}^{\text{Dec}}(\text{ct}_1, n_1)$  and  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$  tightly. Additionally, we utilize the deterministic property to embed the challenge tightly. Define games  $G_0 - G_9$  as in Fig. 7, 8.

GAMES  $G_0 - G_3$ . Games  $G_0 - G_3$  are identical to the  $G_0 - G_3$  in Theorem 1. GAME  $G_4$ . In game  $G_4$ , we abort whenever the adversary did not query  $G(K^*)$  (which is equal to  $\text{HS}^*$ ) but it queried  $H_1(\text{HS}^*, H_T(\text{ct}^*, n^*))$ ,

<sup>5</sup> Applying the simple CPA transform depicted in Fig. 1 to DPKE.

$\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$ <hr/> <pre style="margin: 0; padding: 0;"> 1 : <b>if</b> more than 1 query <math>\vee (\text{ct}_2, n_2) = (\text{ct}^*, n^*)</math> : <b>return</b> <math>\perp</math> 2 : <math>K' \leftarrow \text{decaps}(\text{sk}, \text{ct}_2), \text{HS}' \leftarrow G(K')</math> // <math>G_0 - G_8</math> 3 : <b>if</b> <math>(\text{ct}_2, n_2) = (\text{ct}_1, n_1)</math> : // <math>G_8 -</math> 4 : <b>if</b> <math>\text{guess} = 0</math> : <b>return</b> <math>\perp</math> 5 : <b>if</b> <math>\exists K'</math> s. th. <math>(K', \text{ct}_1, \text{HS}') \in \mathcal{L}</math> : extract <math>\text{HS}'</math> 6 : <b>else</b> <math>\text{HS}' \leftarrow \mathcal{L}_{\text{Dec}} = \{\text{ct}_1, \text{HS}'\}</math> 7 : <b>if</b> <math>(\text{ct}_2, n_2) \neq (\text{ct}_1, n_1) \vee \mathcal{A}</math> did not query <math>\mathcal{O}^{\text{Dec}}</math> before : // <math>G_9</math> 8 : <b>if</b> <math>\text{guess}' = 0</math> : <b>return</b> <math>\perp</math> 9 : <b>if</b> <math>\exists K'</math> s. th. <math>(K', \text{ct}_2, \text{HS}) \in \mathcal{L}</math> : extract <math>\text{HS}'</math> 10 : <b>else</b> <math>\text{HS}' \leftarrow_{\\$} \{0, 1\}^n, \mathcal{L}_{\text{Dec}}^{\text{MAC}} = \{\text{ct}_2, \text{HS}'\}</math> 11 : <math>\text{SHTS} \leftarrow H_2(\text{HS}', H_T(\text{ct}, n)), f_{k_S} \leftarrow H_4(\text{SHTS})</math> 12 : <b>if</b> <math>\text{SHTS} = \text{SHTS}_b</math> : <b>abort</b> // <math>G_2 -</math> 13 : <b>if</b> <math>\text{MAC.Vrf}(f_{k_S}, \text{txt}, \text{tag}) = \text{true}</math>: 14 : <b>if</b> <math>\mathcal{A}</math> did not query <math>H_4(\text{SHTS})</math> : <b>abort</b> // <math>G_2 -</math> 15 : <b>if</b> <math>\mathcal{A}</math> did not query <math>H_2(\text{HS}', H_T(\text{ct}_2, n_2))</math>: <b>abort</b> // <math>G_3 -</math> 16 : <b>return</b> <math>\text{HS}'</math> 17 : <b>return</b> <math>\perp</math> </pre> <hr/> $G(K) \quad // \quad G_6 -$ <hr/> <pre style="margin: 0; padding: 0;"> 1 : <b>if</b> <math>\exists \text{HS}</math> s.th. <math>(K, \text{HS}) \in \mathcal{L}_G</math> : <b>return</b> <math>\text{HS}</math> 2 : <math>\text{ct} = \text{enc}'(\text{pk}, K), \text{HS} \leftarrow_{\\$} \{0, 1\}^n</math> 3 : <b>if</b> <math>\text{ct} = \text{ct}_1</math> : <math>\text{HS}' \leftarrow \mathcal{L}_{\text{Dec}} = \{\text{ct}_1, \text{HS}'\}, \text{HS} = \text{HS}'</math> // <math>G_8, G_9</math> 4 : <b>if</b> <math>\text{ct} = \text{ct}_2</math> : <math>\text{HS}' \leftarrow \mathcal{L}_{\text{Dec}}^{\text{MAC}} = \{\text{ct}_2, \text{HS}'\}, \text{HS} = \text{HS}'</math> // <math>G_9</math> 5 : <math>\mathcal{L}_G = \mathcal{L}_G \cup \{K, \text{HS}\}, \mathcal{L} = \mathcal{L} \cup \{K, \text{ct}, \text{HS}\}</math> 6 : <b>return</b> <math>\text{HS}</math> </pre>
--

**Fig. 8.** Games for the proof of Theorem 3

$H_2(\text{HS}^*, H_T(\text{ct}^*, n^*))$  or  $H_3(\text{HS}^*)$ . Similar to Theorem 1, the probability that  $\mathcal{A}$  queries  $\text{HS}^*$  to  $H_1, H_2$  or  $H_3$  is upper bounded by  $\frac{q_{H_1} + q_{H_2} + q_{H_3}}{2^n}$  and hence we have

$$|\Pr[G_3^{\mathcal{A}} \Rightarrow 1] - \Pr[G_4^{\mathcal{A}} \Rightarrow 1]| \leq \frac{q_{H_1} + q_{H_2} + q_{H_3}}{2^n}.$$

GAME  $G_5$ . In game  $G_5$ ,  $(\text{CHTS}_0^*, \text{SHTS}_0^*, \text{dHS}_0^*)$  is replaced by  $(\text{CHTS}_0^*, \text{SHTS}_0^*, \text{dHS}_0^*) \leftarrow \{0, 1\}^{3n}$ . Define QUERY as the event that  $H_1(\text{HS}^*, H_T(\text{ct}^*, n^*))$ ,  $H_2(\text{HS}^*, H_T(\text{ct}^*, n^*))$ , or  $H_3(\text{HS}^*)$  is queried by the adversary. Then,  $G_5$  is identical to  $G_4$  in  $\mathcal{A}$ 's view unless the event QUERY happens. Thus, we have

$$|\Pr[G_4^{\mathcal{A}} \Rightarrow 1] - \Pr[G_5^{\mathcal{A}} \Rightarrow 1]| \leq \Pr[\text{QUERY} : G_5].$$

One can see that in  $G_5$ , the bit  $b$  is independent of  $\mathcal{A}$ 's view, thus

$$\Pr[G_5^{\mathcal{A}} \Rightarrow 1] = \frac{1}{2}.$$

GAME  $G_6$ . In game  $G_6$ , the challenger simulates the random oracle  $G$  as follows: When adversary  $\mathcal{A}$  queries  $G(K)$ , return  $G(K)$  if it has been previously defined, otherwise randomly select  $\text{HS} \leftarrow_{\$} \{0, 1\}^n$  and return it. We also compute  $\text{ct} = \text{enc}'(\text{pk}, K)$ , and update  $\mathcal{L}_G = \mathcal{L}_G \cup (K, \text{HS})$ ,  $\mathcal{L} = \mathcal{L} \cup (K, \text{ct}, \text{HS})$ . We have

$$\Pr[\text{QUERY} : G_5] = \Pr[\text{QUERY} : G_6].$$

GAME  $G_7$ . In game  $G_7$ , we define ERO as the event that  $\mathcal{L}_G$  contains an entry  $(K, \text{HS})$  with  $\text{dec}'(\text{sk}, \text{enc}'(\text{pk}, K)) \neq K$ . Upon ERO, we immediately abort. It is noteworthy that GAME  $G_6$  and GAME  $G_7$  exhibit identical distributions when ERO does not occur (as implied by  $\delta$ -correctness). Thus we have

$$\Pr[\text{QUERY} : G_6] \leq \Pr[\text{QUERY} : G_7] + \delta.$$

GAME  $G_8$ . In game  $G_8$ , the challenger simulates the  $\mathcal{O}^{\text{Dec}}(\text{ct}_1, n_1)$  oracle without the secret key. Initially, we randomly choose  $\text{guess} \leftarrow_{\$} \{0, 1\}$  to guess whether  $K' = \text{decaps}(\text{sk}, \text{ct}_1)$  equals to  $\perp$ . If  $\text{guess} = 0$ , we assume  $K' = \perp$ , and thus return  $\perp$  in  $\mathcal{O}^{\text{Dec}}$ . If  $\text{guess} = 1$  and the corresponding  $(\cdot, \text{ct}_1, \cdot) \in \mathcal{L}$ , we directly extract the corresponding  $\text{HS}'$  from  $\mathcal{L}$ . If there's no such  $(\cdot, \text{ct}_1, \cdot)$  in  $\mathcal{L}$ , we can conclude that  $\mathcal{A}$  has not queried  $G(K')$  before based on rigid property of the DPKE. Assuming the adversary has queried  $G(K')$  before, this implies the existence of  $(K', \text{ct}'_1, \cdot) \in \mathcal{L}$ , where  $\text{enc}'(\text{pk}, K') = \text{ct}'_1$ . According to the rigid property, we have  $\text{ct}_1 = \text{ct}'_1$ . This contradicts the condition. Therefore, We just sample a uniformly random value  $\text{HS}' \leftarrow_{\$} \{0, 1\}^n$ , and define  $\mathcal{L}^{\text{Dec}} = (\text{ct}_1, \text{HS}')$ . Finally, we utilize  $\text{HS}'$  to compute  $(\text{tk}_C, \text{tk}_S)$ , thereby perfectly simulating  $\mathcal{O}^{\text{Dec}}$ . To maintain consistency with  $\mathcal{O}^{\text{Dec}}$  and random oracle  $G$ , we change  $G$  as follow. When simulating  $G(K)$  later, first compute  $\text{ct} = \text{enc}'(\text{pk}, K)$ . If  $\text{ct} = \text{ct}_1$ , directly return  $\text{HS}'$  from  $\mathcal{L}_{\text{Dec}}$  in this case. If  $\text{ct} \neq \text{ct}_1$ , we can assert that  $\text{dec}'(\text{sk}, \text{ct}) \neq \text{dec}'(\text{sk}, \text{ct}_1)$  based on the rigid property.

Moreover, if the  $(\text{ct}_2, n_2, \cdot, \cdot)$  input to  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  is equal to the  $(\text{ct}_1, n_1)$  input to  $\mathcal{O}^{\text{Dec}}$ , that is,  $(\text{ct}_1, n_1) = (\text{ct}_2, n_2)$ . If  $\text{guess} = 0$ , we directly return  $\perp$  in  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$ . Else we check if  $(\cdot, \text{ct}_1, \cdot) \in \mathcal{L}$ . If so, we find the corresponding  $\text{HS}'$ . Otherwise, we extract  $\text{HS}'$  from  $\mathcal{L}^{\text{Dec}}$ . Then we can use  $\text{HS}'$  to simulate  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$ . Note that if ERO does not happen, this simulation is perfect when guessing correctly, thus we have

$$\Pr[\text{QUERY} : G_7] = 2\Pr[\text{QUERY} : G_8].$$

GAME  $G_9$ . In game  $G_9$ , the challenger can simulate  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\text{ct}_2, n_2, \text{tag}, \text{txt})$  without the secret key. We modify  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  as follows: If the  $(\text{ct}_2, n_2, \cdot, \cdot)$  input to  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  is not equal to the  $(\text{ct}_1, n_1)$  input to  $\mathcal{O}^{\text{Dec}}$  or  $\mathcal{A}$  did not query  $\mathcal{O}^{\text{Dec}}$  before, we initially make another guess  $\text{guess}' \leftarrow_{\$} \{0, 1\}$  to determine whether  $K' = \perp$ .

If  $guess' = 0$ , we simply return  $\perp$  in  $\mathcal{O}_{MAC}^{Dec}$ . If  $guess' = 1$  and the corresponding  $(\cdot, ct_2, \cdot) \in \mathcal{L}$ , we extract the corresponding  $HS'$ . Otherwise, we sample  $HS' \leftarrow_{\$} \{0, 1\}^n$  and define  $\mathcal{L}_{MAC}^{Dec} = (ct_2, HS')$ . We then utilize  $HS'$  to perfectly simulate  $\mathcal{O}_{MAC}^{Dec}$ . To maintain consistency with  $\mathcal{O}_{MAC}^{Dec}$  and the random oracle  $G$ , when simulating  $G(K)$  later, first compute  $ct = enc'(pk, K)$ . If  $ct = ct_2$ , directly return  $HS'$  from  $\mathcal{L}_{Dec}^{MAC}$ , otherwise, we simulate  $G$  as before. Note that this analysis is identical to  $G_8$ . It is apparent that based on the rigid property of DPKE this simulation is perfect when guessing correctly. Thus, we have

$$\Pr[\text{QUERY} : G_8] = 2 \Pr[\text{QUERY} : G_9].$$

Now, we construct an OW-CPA adversary  $\mathcal{C}(pk, ct^*)$  against the DKEM.  $\mathcal{C}$  samples  $guess, guess', n^*$  as in  $G_9$  and  $\text{CHTS}^*, \text{SHTS}^*, \text{dHS}^* \leftarrow_{\$} \{0, 1\}^{3n}$ . Then  $\mathcal{C}$  picks six  $2q_{H_k}$ -wise ( $k \in \{1, 2, 3, 4, H, T\}$ ) independent functions and runs  $\mathcal{A}^{G, H_i, \mathcal{O}^{Dec}, \mathcal{O}_{MAC}^{Dec}}(pk, ct^*, n^*, \text{CHTS}^*, \text{SHTS}^*, \text{dHS}^*)$  as in game  $G_9$ , lastly selects a  $K'$  from G-List such that  $enc'(K') = ct^*$ , and returns  $K'$ . Note that if ERO does not happen,  $\mathcal{C}$  returns  $K^*$  with probability  $\Pr[\text{QUERY} : G_9]$ . Thus  $\text{Adv}_{DKEM}^{OW-CPA}(\mathcal{C}) \geq \Pr[\text{QUERY} : G_9]$ . Putting everything together, we have

$$\begin{aligned} \text{Adv}_{KEM}^{IND-1CCA-MAC}(\mathcal{A}) &\leq 4\text{Adv}_{DKEM}^{OW-CPA}(\mathcal{C}) + \text{Adv}_{MAC}^{EUF-0T}(\mathcal{B}) \\ &\quad + \frac{q_{H_1} + 2q_{H_2} + q_{H_3} + q_{H_D} + q_{H_T} + 6}{2^n} + \delta \end{aligned}$$

□

**Theorem 4.** *Let  $KEM = (\text{gen}, \text{encaps}, \text{decaps}) = T(\text{DPKE}(\text{gen}', \text{enc}', \text{dec}'))$  be a DKEM and the underlying  $\delta$ -correctness DPKE is rigid. Let the KDFs and the hash function in the IND-1CCA-MAC\* game be modeled as random oracles. Then, for any PPT adversary  $\mathcal{A}$  making at most  $q_G, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{H_D}, q_{H_T}$  queries to  $G, H_1, H_2, H_3, H_4, H_D, H_T$  respectively, there exists an OW-CPA adversary  $\mathcal{C}$  and an EUF-0T adversary  $\mathcal{B}$  such that*

$$\begin{aligned} \text{Adv}_{KEM}^{IND-1CCA-MAC^*}(\mathcal{A}) &\leq 2\text{Adv}_{DKEM}^{OW-CPA}(\mathcal{C}) + \text{Adv}_{MAC}^{EUF-0T}(\mathcal{B}) \\ &\quad + \frac{q_{H_1} + 2q_{H_2} + q_{H_3} + q_{H_D} + q_{H_T} + 6}{2^n} + \delta \end{aligned}$$

where  $\mathcal{C}$  have approximately the same running time as  $\mathcal{A}$ .

*Proof.* It is easy to see that except for  $G_9$ , the proof is the same as the proof of the Theorem 3. In the proof of IND-1CCA-MAC\*, we can simply define  $G_9$  to be identical to  $G_8$ . Thus, in this proof we have  $\Pr[\text{QUERY} : G_8] = \Pr[\text{QUERY} : G_9]$ . □

### 3.2 OW-CPA/IND-CPA/D-OW-CPA KEMs Imply IND-1CCA-MAC\* in the QROM

We now prove that any OW-CPA/IND-CPA/D-OW-CPA KEMs are also IND-1CCA-MAC\* secure in the QROM with an EUF-0T secure MAC.

**Theorem 5.** *Let  $\text{KEM} = (\text{gen}, \text{encaps}, \text{decaps})$  be a KEM. Let the KDFs and the hash function in the  $\text{IND-1CCA-MAC}^*$  game be modeled as quantum random oracles. Then, for any PPT adversary  $\mathcal{A}$  issuing at most one single (classical) query to the  $\mathcal{O}^{\text{Dec}}$ ,  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  oracle and making at most  $q_G, q_{H_1}, q_{H_2}, q_{H_3}, q_{H_4}, q_{H_D}, q_{H_T}$  queries to  $G, H_1, H_2, H_3, H_4, H_D, H_T$  respectively and let  $q_{123} = q_{H_1} + q_{H_2} + q_{H_3} + 1$ , there exists a PPT OW-CPA adversary  $\mathcal{C}$ , a PPT D-OW-CPA adversary  $\hat{\mathcal{C}}$  (if KEM is DPKE), a PPT IND-CPA adversary  $\mathcal{D}$ , a PPT EUF-0T adversary  $\mathcal{B}_1$  and a PPT EUF-0T adversary  $\mathcal{B}_2$  such that*

$$\begin{aligned} & \text{Adv}_{\text{KEM}}^{\text{IND-1CCA-MAC}^*}(\mathcal{A}) \\ & \leq \frac{2q_{123} + 6q_{H_4}(q_{H_D} + 2)}{2^{n/2}} + \frac{(q_{H_T} + 4)^2 + (q_{H_2} + 1)^2}{2^n} + \text{Adv}_{\text{MAC}}^{\text{EUF-0T}}(\mathcal{B}_1) \\ & \quad + 8q_G(q_{H_1} + q_{H_2} + 1). \\ & \quad \sqrt{\text{Adv}_{\text{KEM}}^{\text{OW-CPA}}(\mathcal{C}) + \frac{1}{2^{2n}} + 6q_{H_4}(q_{H_D} + 2)2^{-n/2} + \text{Adv}_{\text{MAC}}^{\text{EUF-0T}}(\mathcal{B}_2)}. \end{aligned}$$

$$\begin{aligned} & \text{Adv}_{\text{KEM}}^{\text{IND-1CCA-MAC}^*}(\mathcal{A}) \\ & \leq \frac{2q_{123} + 6q_{H_4}(q_{H_D} + 2)}{2^{n/2}} + \frac{(q_{H_T} + 4)^2 + (q_{H_2} + 1)^2}{2^n} + \text{Adv}_{\text{MAC}}^{\text{EUF-0T}}(\mathcal{B}_1) \\ & \quad + 8(q_{H_1} + q_{H_2} + 1). \\ & \quad \sqrt{\text{Adv}_{\text{DKEM}}^{\text{OW-CPA}}(\hat{\mathcal{C}}) + \frac{1}{2^{2n}} + \delta + 6q_{H_4}(q_{H_D} + 2)2^{-n/2} + \text{Adv}_{\text{MAC}}^{\text{EUF-0T}}(\mathcal{B}_2)}. \end{aligned}$$

$$\begin{aligned} & \text{Adv}_{\text{KEM}}^{\text{IND-1CCA-MAC}^*}(\mathcal{A}) \\ & \leq \frac{2q_{123} + 6q_{H_4}(q_{H_D} + 2)}{2^{n/2}} + \frac{(q_{H_T} + 4)^2 + (q_{H_2} + 1)^2}{2^n} + \text{Adv}_{\text{MAC}}^{\text{EUF-0T}}(\mathcal{B}_1) \\ & \quad + 8(q_{H_1} + q_{H_2} + 1). \\ & \quad \sqrt{2\text{Adv}_{\text{KEM}}^{\text{IND-CPA}}(\mathcal{D}) + \frac{(q_G + 1)^2}{|\mathcal{K}|} + 6q_{H_4}(q_{H_D} + 3)2^{-n/2} + \text{Adv}_{\text{MAC}}^{\text{EUF-0T}}(\mathcal{B}_2)}. \end{aligned}$$

where  $\mathcal{B}_1, \mathcal{B}_2, \mathcal{C}$ , and  $\mathcal{D}$  have approximately the same running time as  $\mathcal{A}$ .

**Proof Sketch:** The proof consists of four main steps. The first is embedding the underlying hard problem by replacing the real key  $\text{HS}^*$  with a random value. When the underlying KEM is OW-CPA-secure, we use general OW2H [3] to argue the reprogramming impact. When the KEM is IND-CPA-secure or D-OW-CPA-secure, we use the double-sided OW2H [7] to discuss the impact of reprogramming. Since the embedded IND-CPA game is decisional, we also need to argue the advantage that searching for a reprogramming point results in a double-sided oracle.

The second step is simulate the  $\mathcal{O}^{\text{Dec}}(\overline{ct}, \overline{n})$  oracle without the secret key. Initially, we utilize an internal hash function  $H_{12} = (H_1, H_2)$  to simulate the quantum random oracles  $H_1$  and  $H_2$ . We first randomly choose  $guess \leftarrow_{\$} \{0, 1\}$  to guess whether  $\text{decaps}(\text{sk}, \overline{ct}) = \perp$ . When  $guess = 0$ , we return  $\perp$ . When  $guess = 1$ , we proceed with the simulation as follows. As discussed in Sect. 1.3, we directly reprogram  $H_{12}(\overline{HS}, \overline{t})$  to random values  $\Theta = (\overline{chts}, \overline{shts})$ , where  $\overline{HS} = G(K)$  and  $\overline{t} = H_T(\overline{ct}, \overline{n})$ , then output  $H_D(\overline{chts}), H_D(\overline{shts})$  in the  $\mathcal{O}^{\text{Dec}}$ . Intuitively, the simulation is perfect if we reprogram  $H_{12}(\overline{HS}, \overline{t})$  to  $\Theta$  when the adversary  $\mathcal{A}$  first queries  $(\overline{HS}, \overline{t})$  to either  $H_1$  or  $H_2$  in the ROM. However, in the QROM, it is hard to define when the adversary makes a query  $(\overline{HS}, \overline{t})$  to  $H_1$  or  $H_2$ . Therefore, in the QROM, we use the idea from [21] to argue it differently. As discussed in Sect. 1.3 we find that the simulation is perfect if the predicate  $\mathcal{O}^{\text{Dec}}(\overline{ct}, \overline{n}) = (H_D(H_1(\overline{HS}, \overline{t})), H_D(H_2(\overline{HS}, \overline{t})))$  is satisfied. Since in the practical implementation of the  $\mathcal{O}^{\text{Dec}}(\overline{ct}, \overline{n})$  oracle, there is an implicit classical query to  $H_{12}$ , which is removed during the oracle's simulation in  $\mathcal{O}^{\text{Dec}}$ . Therefore, this specific query cannot be measured. Hence, we employ the refined optional-query measure-and-reprogram technique [21] to argue the simulation impact.

The third step is simulate the  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}(\overline{ct}, \overline{n}, \text{txt}, \text{tag})$  oracle without the secret key. Recall that  $(\overline{ct}, \overline{n})$  remains consistent with the  $\mathcal{O}^{\text{Dec}}$  oracle according to the definition in IND-1CCA-MAC\*. When the  $guess$  value in  $\mathcal{O}^{\text{Dec}}$  is 0, we return  $\perp$  in  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$ . Otherwise, we proceed with the following simulation. While simulating  $\mathcal{O}^{\text{Dec}}(\overline{ct}, \overline{n})$ , we employ the refined optional-query measure and reprogram technique on  $H_{12}$ , where the measurement yields  $x$ , and then reprogram  $H_{12}$  on  $x$  to  $\Theta$ . If  $\text{MAC.Vrf}(fk_S, \text{txt}, \text{tag})$  is correct, we need to return HS in  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$ , where  $fk_S = H_D(\overline{shts})$ . Intuitively, we can directly use  $x$  (which includes HS) to output HS. However, it is crucial to discuss the order of  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  query and the measurement in the refined optional-query measure and reprogram technique. If  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  query occurs after the measurement, we can utilize  $x$  to perfectly simulate it by return HS. However, if  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  query occurs before the measurement, we cannot return HS since we have not yet measured it to obtain  $x$ . Nonetheless, this scenario implies that all  $H_{12}$  queries made by the adversary so far are not reprogrammed and unrelated to the reprogrammed values  $\Theta$ , which are utilized to derive the MAC key. Thus, we can argue that the adversary  $\mathcal{A}$  cannot distinguish  $fk_S$  from a random value using OW2H Lemma. Hence, according to the security of MAC, the adversary cannot forge a valid tag. Therefore, if  $\mathcal{O}_{\text{MAC}}^{\text{Dec}}$  query occurs before the measurement, we directly output  $\perp$ .

Finally, we reprogram  $(\text{CHTS}_0^*, \text{SHTS}_0^*, \text{dHS}_0^*) = (H_1(\text{HS}^*, t^*), H_2(\text{HS}^*, t^*), H_3(\text{HS}^*))$  to random values. In this case, the adversary's probability of winning this game is  $1/2$ . We then utilize  $\text{HS}^*$  to be indistinguishable from random for the adversary to analyze the impact of such reprogramming using OW2H Lemma.

*Proof.* We provide the complete proof in complete version of the paper.  $\square$

*Remark 2.* One can demonstrate the security of the IND-1CCA-MAC based on CPA-secure KEMs in the QROM using a similar proof technique as Theorem 5 with greater loss reduction compared to IND-1CCA-MAC\* because it is necessary to utilize the Measure-and-Reprogram technique twice.



### 3.3 Multi-stage Security for TLS 1.3 from IND-1CCA-MAC\*

We now demonstrate that if the IND-1CCA-MAC\* is satisfied, the 1-RTT TLS 1.3 handshake is secure in the MultiStage model. Theorem 6 aligns with Theorem 5 in [20], but substitutes the IND-1CCA-MAC with the IND-1CCA-MAC\*.

**Theorem 6.** *The TLS 1.3 full 1-RTT handshake is secure in the MultiStage model if the underlying KEM is IND-1CCA-MAC\* (and the signature is secure). Formally for any Multi-Stage PPT adversary  $\mathcal{A}$ , there exist PPT adversaries  $\{\mathcal{B}_i\}_{i \in [6]}$  such that*

$$\text{Adv}_{\text{TLS1.3-1RTT}}^{\text{multi-stage}}(\mathcal{A}) \leq 6t_s \left( \text{Adv}_{\text{H}}^{\text{coll}}(\mathcal{B}_1) + t_u \text{Adv}_{\text{Sig}}^{\text{euf-cma}}(\mathcal{B}_2) + t_s \left( \text{Adv}_{\text{KEM}}^{\text{ind-1cca-mac}^*}(\mathcal{B}_3) + 2\text{Adv}_{\text{HKDF.Exp}}^{\text{prf}}(\mathcal{B}_4) + \text{Adv}_{\text{HKDF.Ext}}^{\text{prf}}(\mathcal{B}_5) + \text{Adv}_{\text{HKDF.Exp}}^{\text{prf}}(\mathcal{B}_6) \right) \right)$$

where  $t_s$  (resp.  $t_u$ ) is the maximal number of sessions (resp. users).

**Proof Sketch:** This proof is identical to the proof in [20], Theorem 5, except for replacing IND-1CCA-MAC with IND-1CCA-MAC\*. For detailed steps, please refer to the original proof in [15], Theorem 6.4. They utilized the snPRF-ODH assumption to substitute HS with a random  $\widetilde{\text{HS}}$  [15]. Following [20], we directly apply the IND-1CCA-MAC\* to concurrently replace CHTS, SHTS, and dHS with random values, and subsequently leverages PRF properties to replace additional keys with random values. In particular, IND-1CCA-MAC\* is sufficient for the security of TLS 1.3. The complete proof is provided in complete version of the paper.

**Theorem 7.** *The modified TLS 1.3 handshake in the pre-shared key (optional) 0-RTT mode with key exchange (i.e., TLS 1.3 PSK-(EC)-DHE 0-RTT) is secure in the MultiStage model if the underlying KEM is IND-1CCA-MAC\* (and signature, MAC, etc. are secure), in the sense of Dowling et al. [15].*

We provide the complete statement and a proof sketch for Theorem 7 in complete version of the paper.

*Remark 3.* Combining Theorem 2, 4, 5, 6, 7, we obtain the security proof of TLS 1.3 from OW-CPA/IND-CPA/D-OW-CPA KEMs (with a secure MAC) in the ROM and QROM.

**Acknowledgements.** We would like to thank anonymous reviewers of Asiacrypt 2024 for their insightful comments and suggestions. We thank Jieyu Zheng for her help in benchmarking. Biming Zhou and Yunlei Zhao was supported by the National Key R&D Program of China (No. 2022YFB2701601), General Project of State Key Laboratory of Cryptography (No. MMKFKT202227), Technical Standard Project of Shanghai Scientific and Technological Committee (No. 21DZ2200500), Shanghai Collaborative Innovation Fund (No. XTCX-KJ-2023-54), and Special Fund for Key Technologies in Blockchain of Shanghai Scientific and Technological Committee (No. 23511100300). Haodong Jiang was supported by the National Key R&D Program of China (No. 2021YFB3100100), and the National Natural Science Foundation of China (No. 62002385).

## References

1. Open-quantum-safe `openssl`. <https://github.com/open-quantum-safe/openssl> (2024)
2. Albrecht, M.R., Bernstein, D.J., Chou, T., Cid, C., Gilcher, J., Lange, T., Maram, V., von Maurich, I., Misoczki, R., Niederhagen, R., Paterson, K.G., Persichetti, E., Peters, C., Schwabe, P., Sendrier, N., Szefer, J., Tjhai, C.J., Tomlinson, M., Wang, W.: Classic mceliece. Technical report, National Institute of Standards and Technology (2020), <https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>
3. Ambainis, A., Hamburg, M., Unruh, D.: Quantum security proofs using semi-classical oracles. In: Boldyreva, A., Micciancio, D. (eds.) *Advances in Cryptology – CRYPTO 2019, Part II*. Lecture Notes in Computer Science, vol. 11693, pp. 269–295. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2019)
4. Angel, Y., Dowling, B., Hülsing, A., Schwabe, P., Weber, F.J.: Post quantum noise. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) *ACM CCS 2022: 29th Conference on Computer and Communications Security*. pp. 97–109. ACM Press, Los Angeles, CA, USA (Nov 7–11, 2022)
5. Azouaoui, M., Bronchain, O., Hoffmann, C., Kuzovkova, Y., Schneider, T., Standardt, F.X.: Systematic study of decryption and re-encryption leakage: The case of kyber. In: Balasch, J., O’Flynn, C. (eds.) *COSADE 2022: 13th International Workshop on Constructive Side-Channel Analysis and Secure Design*. Lecture Notes in Computer Science, vol. 13211, pp. 236–256. Springer, Heidelberg, Germany, Leuven, Belgium (Apr 11–12, 2022)
6. Bernstein, D.J., Persichetti, E.: Towards kem unification. *IACR Cryptol. ePrint Arch, Report 2018/526* (2018), <https://eprint.iacr.org/2018/526.pdf>
7. Bindel, N., Hamburg, M., Hövelmanns, K., Hülsing, A., Persichetti, E.: Tighter proofs of CCA security in the quantum random oracle model. In: Hofheinz, D., Rosen, A. (eds.) *TCC 2019: 17th Theory of Cryptography Conference, Part II*. Lecture Notes in Computer Science, vol. 11892, pp. 61–90. Springer, Heidelberg, Germany, Nuremberg, Germany (Dec 1–5, 2019)
8. Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Stehlé, D.: Crystals - kyber: A cca-secure module-lattice-based kem. 2018 IEEE European Symposium on Security and Privacy (EuroS&P) pp. 353–367 (2017)
9. Brendel, J., Fiedler, R., Günther, F., Janson, C., Stebila, D.: Post-quantum asynchronous deniable key exchange and the Signal handshake. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) *PKC 2022: 25th International Conference on Theory and Practice of Public Key Cryptography, Part II*. Lecture Notes in Computer Science, vol. 13178, pp. 3–34. Springer, Heidelberg, Germany, Virtual Event (Mar 8–11, 2022)
10. Brendel, J., Fischlin, M., Günther, F., Janson, C.: PRF-ODH: Relations, instantiations, and impossibility results. In: Katz, J., Shacham, H. (eds.) *Advances in Cryptology – CRYPTO 2017, Part III*. Lecture Notes in Computer Science, vol. 10403, pp. 651–681. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2017)

11. Danba, O., Hoffstein, J., Hulsing, A., Rijneveld, J., Schanck, J.M., Schwabe, P., Whyte, W., Zhang, Z., Saito, T., Yamakawa, T., Xagawa, K.: Ntru. Technical report, National Institute of Standards and Technology (2020), <https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>
12. Don, J., Fehr, S., Majenz, C.: The measure-and-reprogram technique 2.0: Multi-round fiat-shamir and more. In: Micciancio, D., Ristenpart, T. (eds.) *Advances in Cryptology – CRYPTO 2020, Part III*. Lecture Notes in Computer Science, vol. 12172, pp. 602–631. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2020)
13. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the Fiat-Shamir transformation in the quantum random-oracle model. In: Boldyreva, A., Micciancio, D. (eds.) *Advances in Cryptology – CRYPTO 2019, Part II*. Lecture Notes in Computer Science, vol. 11693, pp. 356–383. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2019)
14. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Online-extractability in the quantum random-oracle model. In: Dunkelman, O., Dziembowski, S. (eds.) *Advances in Cryptology – EUROCRYPT 2022, Part III*. Lecture Notes in Computer Science, vol. 13277, pp. 677–706. Springer, Heidelberg, Germany, Trondheim, Norway (May 30 – Jun 3, 2022)
15. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the TLS 1.3 handshake protocol. *Journal of Cryptology* 34(4), 37 (Oct 2021)
16. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) *Advances in Cryptology – CRYPTO’99*. Lecture Notes in Computer Science, vol. 1666, pp. 537–554. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999)
17. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. *Journal of Cryptology* 26(1), 80–101 (Jan 2013)
18. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) *TCC 2017: 15th Theory of Cryptography Conference, Part I*. Lecture Notes in Computer Science, vol. 10677, pp. 341–371. Springer, Heidelberg, Germany, Baltimore, MD, USA (Nov 12–15, 2017)
19. Hövelmanns, K., Kiltz, E., Schäge, S., Unruh, D.: Generic authenticated key exchange in the quantum random oracle model. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*. Lecture Notes in Computer Science, vol. 12111, pp. 389–422. Springer, Heidelberg, Germany, Edinburgh, UK (May 4–7, 2020)
20. Huguenin-Dumittan, L., Vaudenay, S.: On IND-qCCA security in the ROM and its applications - CPA security is sufficient for TLS 1.3. In: Dunkelman, O., Dziembowski, S. (eds.) *Advances in Cryptology – EUROCRYPT 2022, Part III*. Lecture Notes in Computer Science, vol. 13277, pp. 613–642. Springer, Heidelberg, Germany, Trondheim, Norway (May 30 – Jun 3, 2022)
21. Jiang, H., Ma, Z., Zhang, Z.: Post-quantum security of key encapsulation mechanism against CCA attacks with a single decapsulation query. In: Guo, J., Steinfeld, R. (eds.) *Advances in Cryptology – ASIACRYPT 2023, Part IV*. Lecture Notes in Computer Science, vol. 14441, pp. 434–468. Springer, Heidelberg, Germany, Guangzhou, China (Dec 4–8, 2023)

22. Jiang, H., Zhang, Z., Chen, L., Wang, H., Ma, Z.: IND-CCA-secure key encapsulation mechanism in the quantum random oracle model, revisited. In: Shacham, H., Boldyreva, A. (eds.) *Advances in Cryptology – CRYPTO 2018, Part III. Lecture Notes in Computer Science*, vol. 10993, pp. 96–125. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2018)
23. Jiang, H., Zhang, Z., Ma, Z.: Key encapsulation mechanism with explicit rejection in the quantum random oracle model. In: Lin, D., Sako, K. (eds.) *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II. Lecture Notes in Computer Science*, vol. 11443, pp. 618–645. Springer, Heidelberg, Germany, Beijing, China (Apr 14–17, 2019)
24. Jiang, H., Zhang, Z., Ma, Z.: Tighter security proofs for generic key encapsulation mechanism in the quantum random oracle model. In: Ding, J., Steinwandt, R. (eds.) *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*. pp. 227–248. Springer, Heidelberg, Germany, Chongqing, China (May 8–10, 2019)
25. Kuchta, V., Sakzad, A., Stehlé, D., Steinfeld, R., Sun, S.: Measure-rewind-measure: Tighter quantum random oracle model proofs for one-way to hiding and CCA security. In: Canteaut, A., Ishai, Y. (eds.) *Advances in Cryptology – EUROCRYPT 2020, Part III. Lecture Notes in Computer Science*, vol. 12107, pp. 703–728. Springer, Heidelberg, Germany, Zagreb, Croatia (May 10–14, 2020)
26. Naehrig, M., Alkim, E., Bos, J.W., Ducas, L., Easterbrook, K., LaMacchia, B., Longa, P., Mironov, I., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D.: Frodokem learning with errors key encapsulation. <https://frodokem.org/files/FrodoKEM-specification-20210604.pdf> (2021)
27. National Institute for Standards and Technology: Post-quantum cryptography project (2022), <https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions>
28. National Institute of Standards and Technology: Module-lattice-based key-encapsulation mechanism standard. FIPS203 (Aug 2023), initial Public Draft
29. Paquin, C., Stebila, D., Tamvada, G.: Benchmarking post-quantum cryptography in TLS. In: Ding, J., Tillich, J.P. (eds.) *Post-Quantum Cryptography - 11th International Conference, PQCrypto 2020*. pp. 72–91. Springer, Heidelberg, Germany, Paris, France (Apr 15–17, 2020)
30. Schwabe, P., Stebila, D., Wiggers, T.: Post-quantum TLS without handshake signatures. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) *ACM CCS 2020: 27th Conference on Computer and Communications Security*. pp. 1461–1480. ACM Press, Virtual Event, USA (Nov 9–13, 2020)
31. Schwabe, P., Stebila, D., Wiggers, T.: More efficient post-quantum KEMTLS with pre-distributed public keys. In: Bertino, E., Shulman, H., Waidner, M. (eds.) *ESORICS 2021: 26th European Symposium on Research in Computer Security, Part I. Lecture Notes in Computer Science*, vol. 12972, pp. 3–22. Springer, Heidelberg, Germany, Darmstadt, Germany (Oct 4–8, 2021)
32. Ueno, R., Xagawa, K., Tanaka, Y., Ito, A., Takahashi, J., Homma, N.: Curse of re-encryption: A generic power/em analysis on post-quantum kems. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2022, 296–322 (2021)

33. Zhandry, M.: Secure identity-based encryption in the quantum random oracle model. In: Safavi-Naini, R., Canetti, R. (eds.) *Advances in Cryptology – CRYPTO 2012*. Lecture Notes in Computer Science, vol. 7417, pp. 758–775. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012)
34. Zhandry, M.: How to record quantum queries, and applications to quantum indistinguishability. In: Boldyreva, A., Micciancio, D. (eds.) *Advances in Cryptology – CRYPTO 2019, Part II*. Lecture Notes in Computer Science, vol. 11693, pp. 239–268. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2019)



# Post-quantum Asynchronous Remote Key Generation for FIDO2

Jacqueline Brendel<sup>(✉)</sup>, Sebastian Clermont, and Marc Fischlin

Technische Universität Darmstadt, Darmstadt, Germany  
mail@jbrendel-info.de, {sebastian.clermont,marc.fischlin}@tu-darmstadt.de

**Abstract.** The Fast IDentity Online (FIDO) Alliance has developed the widely adopted FIDO2 protocol suite that allows for passwordless online authentication. Cryptographic keys stored on a user's device (e.g. their smartphone) are used as credentials to authenticate to services by performing a challenge-response protocol. Yet, this approach leaves users unable to access their accounts in case their authenticator is lost.

The device manufacturer Yubico thus proposed a FIDO2-compliant mechanism that allows to easily create backup authenticators. Frymann et al. (CCS 2020) have first analyzed the cryptographic core of this proposal by introducing the new primitive of *Asynchronous Remote Key Generation* (ARKG) and accompanying security definitions. Later works instantiated ARKG both from classical and post-quantum assumptions (ACNS 2023, EuroS&P 2023).

As we will point out in this paper, the security definitions put forward and used in these papers do not adequately capture the desired security requirements in FIDO2-based authentication and recovery. This issue was also identified in independent and concurrent work by Stebila and Wilson (AsiaCCS 2024), who proposed a new framework for the analysis of account recovery mechanisms, along with a secure post-quantum instantiation from KEMs and key-blinding signature schemes.

In this work, we propose alternative security definitions for the primitive ARKG when used inside an account recovery mechanism in FIDO2. We give a secure instantiation from KEMs and standard signature schemes, which may in particular provide post-quantum security. Our solution strikes a middle ground between the compact, but (for this particular use case) inadequate security notions put forward by Frymann et al., and the secure, but more involved and highly tailored model introduced by Stebila and Wilson.

**Keywords:** FIDO2 · post-quantum · account recovery · passwordless

## 1 Introduction

FIDO2 encompasses a set of pioneering industry standards for passwordless authentication on the web [5, 14] that is driven and supported by a wide consortium of major industry players and stakeholders. At its core, passwords are

replaced by a sophisticated combination of public-key cryptography and hardware devices such as smartphones or dedicated security keys. In a FIDO2 authentication ceremony, users authenticate themselves by proving in a challenge-response fashion that they control the secret key of a previously registered public-key credential.

There are three main actors: the *authenticator*, the *client*, and the *relying party*. The relying party is the service that a user wants to log in to, i.e., the party that will verify the user's identity. The authenticator is the device holding the secret key material used for signing the cryptographic authentication challenge issued by the relying party. The client is the device that the user uses to access the authenticator and to communicate with the relying party.

For example, suppose a user is logging into their Gmail account using their laptop with a Yubico security key. In that case, their Yubikey is the authenticator, the laptop (and the browser on it) is the client, and Google is the relying party. The set of cryptographic information required by a user for a successful authentication is referred to as a *passkey* [21] or simply a (FIDO2) credential.

### 1.1 The Need for Credential Recovery

When using devices such as a smartphone or a USB security key for authentication, the possibility of transferring or recovering credentials must be considered. Essentially, there are two relevant scenarios:

1. One is the graceful transition from an old authenticator to a new one when the previously registered credentials are still accessible to the user, e.g., when switching smartphones.
2. The other, more challenging, situation occurs when the original credentials are no longer available, e.g., in case the smartphone is lost, stolen, or broken.

While the first scenario can be easily accommodated by logging on with the old device and then adding the new device as an authenticator, the second case is much trickier.

For the secure recovery of *symmetric* secrets, mechanisms have already been created. Signal's *Secure Value Recovery* or Apple's *Secure iCloud Keychain Recovery* serve as two examples. These mechanisms rely on a low-entropy recovery PIN created by the user, which encrypts a high-entropy key stored on a distributed hardware security module (HSM). This high-entropy key is then used to encrypt the user's secrets. Recovery of the high-entropy key is governed by the HSM, which only allows for a limited amount of retries to prevent brute-force attacks.

For the secure recovery of *asymmetric secrets*, as they are used within FIDO2, the aforementioned approach should not, and often cannot, be applied. For once, the entire security relies on a low-entropy PIN chosen by the user. Furthermore, the approach requires trust in cloud vendors and their HSMs to behave as promised, which is an unsatisfactory assumption at best. Lastly, dedicated hardware authenticators, such as FIDO2 security keys, usually generate their

cryptographic secrets locally and do not allow for their extraction from the device.

The FIDO2 Alliance suggests the usage of so-called *multi-device credentials* [21] to ensure reliable access to relying parties. Multi-device credentials are not constrained to the physical devices they were generated on but can be copied across a user's devices to ensure a consistent login experience. While this is an easy-to-use approach, it has numerous security drawbacks.

In principle, it is possible for users to then have multiple, functionally-identical authenticators, by synchronizing the passkeys over a vendor's cloud service. This solves the recovery problem straightforwardly: If one authenticator is lost, a new one can be acquired, and the multi-device credentials can be copied to this new authenticator. However, this also means that:

- The fine-grained revocation of credentials is impossible. All authenticators behave identically and use the same secrets, i.e., a user cannot revoke access for the lost credential, since the credentials on his new authenticator are identical.
- Hardware binding is impossible. A user can never be certain of being the sole owner of their secret keys and thus their account access, as it might have been copied to a different device without their knowledge or consent.

Furthermore, the relying party may insist on single-device passkeys for high-security use cases. This functionality is specified as part of the FIDO2 protocol suite and is thus incompatible with this recovery approach. Lastly, as already mentioned above, dedicated hardware security keys are built to be *tamperproof* which in particular means that they may not support the export of secret key material.

## 1.2 Introducing Backup Authenticators

In light of this, Yubico, one of the leading manufacturers of hardware security devices, takes the stance that allowing secret key material to leave the authenticators is an inherent weakness in the system and must be avoided [13]. Consequently, Yubico does not support the creation of multi-device FIDO2 passkeys and strictly follows a one-device, one-credential policy.

Yubico has thus suggested an alternative approach, using so-called *backup authenticators* (BA) to facilitate account recovery when a user is faced with lost or broken authenticators [16]. This solution is an easy-to-implement extension to the WebAuthn protocol in FIDO2 and bridges the gap in account recovery mechanisms, as it eliminates the need for trusted third parties, while also maintaining the full autonomy of the authenticators.

A backup authenticator is a hardware device that is initialized *once* by creating a cryptographic key pair. The public key of this key pair is then transferred once to the *primary authenticator* (PA), which is a regular authenticator that is used for day-to-day authentications to relying parties. After this initial pairing between backup and primary authenticator, the backup can be stored offline in a



safe location and will only be required in case a user needs to recover credentials, e.g., when they have lost their primary authenticator.

In particular, when registering a new credential with a relying party, and in any subsequent login, the user only needs their primary authenticator. To enable this some additional protected recovery information is stored at the relying party when the FIDO2 credential is first registered. The backup authenticator uses its secret key to recover the authentication data from this externally stored recovery information.

In more detail, the solution by Yubico is a Diffie–Hellman-based protocol, which roughly lets the backup authenticator and the primary authenticator each generate the public part of a Diffie–Hellman (DH) key share. The primary authenticator stores its public DH share as recovery information at the relying party, along with the public key of the joint DH key derived from the PA’s and BA’s DH shares. To serve a recovery request, the backup authenticator computes the joint DH secret by combining the externally stored public DH share of the primary authenticator with its own DH secret. The resulting joint DH secret key can then be used to authenticate the recovery request to the relying party.

Note, that the recovery DH keys are separate from the “regular” FIDO2 credentials, which are generated independently and are used for authentications with the primary authenticator during regular protocol execution.

### 1.3 Security of FIDO2 and Yubico’s Proposal

Backup authenticators and how to create them from the newly introduced primitive for asynchronous remote key generation is not part of the FIDO2 standard. Consequently, there exist no established security requirements, let alone any formal definitions. The cryptographic core of the account recovery extension to WebAuthn has first been formalized and analyzed by Frymann et al. [8] through the introduction of a new cryptographic primitive which they termed *Asynchronous Remote Key Generation*, or ARKG, for short. Since then, a subset of the authors of [8] has also introduced alternative instantiations of the ARKG primitive based on pairings [10] and lattices [7]. The latter makes use of the (equally new) primitive of *split-KEMs*, which was first introduced by Brendel et al. [6] in the context of post-quantum asynchronous key exchange.

As we will point out in more detail shortly, we believe that the ARKG security notions proposed by Frymann et al. [8] are not adequate for the proposed use case of FIDO2 account recovery. In independent and concurrent work to ours, Stebila and Wilson [22], have also arrived at this conclusion. While we opt to “fix” the security notions for ARKG, Stebila and Wilson go a step further and propose a new abstraction for account recovery mechanisms in place of ARKG. Their security analysis is closer to a protocol security analysis than the analysis of a stand-alone cryptographic primitive. As such, no straightforward implications between our security notions and theirs can be shown.

The underlying FIDO2 protocol was first analyzed by Barbosa, Boldyreva, Chen, and Warinschi [1] using the provable security paradigm. A formal analysis

of FIDO2 has been conducted by Guan, Li, He, and Zhao [11]. More comprehensive analyses capturing newer protocol versions and modes, as well as advanced security features have been conducted by Hanzlik, Loss, and Wagner [12] as well as Bindel, Cremers and Zhao [3]. In [4], Bindel, Gama, Guasch, and Ronen analyzed the different attestation modes specified for FIDO2. In particular, [3] proposed and analyzed a full post-quantum instantiation of the latest iteration of the FIDO2 standards CTAP 2.1 and WebAuthn 2, requiring only minor extensions to the protocol, thus showing that the functionality of the FIDO2 protocol family can be achieved without any classical hardness assumptions.

## 1.4 Our Contributions

We have found the security notions for ARKG put forward in [8] and further used in the subsequent works [7, 10] to be a poor fit *when used inside the FIDO2 framework*.<sup>1</sup>

Frymann et al. [8] have defined two security definitions for ARKG. One covers the security of the backup authentication mechanism, called *secret-key security* or *key security*, which intuitively requires that the adversary cannot produce the entire secret key used by honest users to recover their account. This notion comes in slightly different flavors, depending on whether the adversary can communicate with the backup authenticator or not (*strong* vs. *weak*), and whether the adversary needs to attack a given public key or can attempt to fool the relying party with a public key of its choice (*honest* vs. *malicious*).

The second property they propose is *public-key unlinkability*. It captures the relying parties' inability to track users across different services via the backup authenticator's public keys. This is formalized by an indistinguishability notion where the adversary learns a backup authenticator's long-term public key and either receives keys derived from this long-term public key or independently generated keys.

We set off by giving more befitting security notions for both key security and public-key unlinkability, which may also prove to be advantageous in other use cases of ARKG beyond FIDO2. In the course of these adaptations, we change the names of the security properties to *authentication security* and *unlinkability*, since the first property not only protects the secret key but also prevents forgeries, and the privacy property now also takes other available data beyond the public key into account.

In a bit more detail, for authentication security, we strengthen the adversary in line with the FIDO2 use case by counting the adversary already as successful if it can produce signature forgeries under backup keys. In practice, this means an adversary cannot access the recovery mechanism and register new credentials on behalf of the user, locking them out of their accounts.

For unlinkability, we follow a left-or-right approach where the adversary cannot decide which of two backup keys has been generated given derived public

---

<sup>1</sup> We stress that we have not considered other contexts in which ARKG may be employed, for which the originally proposed security properties may be appropriate.

keys and recovery information. This extends the previous definition to now also include the recovery data which is accessible to the adversary in the real-world use case. We will give a detailed discussion of the shortcomings in the analysis of [8] and our adjustments in Sect. 3.

Our second contribution is then the proposal of a FIDO2-compatible ARKG instantiation from standard KEMs and signature schemes in Sect. 4 to provide post-quantum security. The idea is to let the primary authenticator generate the key pair of a signature scheme, encapsulate the key-generating randomness under the backup authenticator’s public key, and store this ciphertext externally at the relying party. During recovery, the backup authenticator can retrieve the randomness and re-generate the signing key.

In particular, we do not rely on the only recently-introduced notions of split KEMs as necessary in [7] or key-blinding signature schemes as in [22]. We note, that both of these primitives, split KEMs and key-blinding signatures, are not by themselves standardized, which may be a requirement for some future implementations and use cases.

## 2 Preliminaries of Asynchronous Remote Key Generation

We start by introducing the notation, terminology, and main definitions used in this paper. In particular, we revisit the definition of Asynchronous Remote Key Generation (ARKG), which was first proposed by Frymann et al. in [8].

*Notation.* We write  $y \leftarrow \text{Alg}(x)$  and  $y \leftarrow \$\text{Alg}(x)$  for the deterministic, resp. probabilistic execution of an algorithm  $\text{Alg}$  on input  $x$  with output  $y$ . We write  $\text{prefix}(x) = y$  to indicate that  $y$  is a prefix of  $x$ . We assume classical algorithms for implementing schemes, with the common notion of *efficiency* if the algorithms run in probabilistic or quantum polynomial time in the length of the security parameter  $\lambda$ , denoted by PPT and QPT, respectively. Explicit randomness is indicated in an algorithm’s input using a semicolon, e.g.,  $\text{Sign}(\text{sk}, m; r)$  denotes the execution of the signing algorithm with randomness  $r$ .

We assume QPT adversaries  $\mathcal{A}$ . Since the honest parties use classical algorithms,  $\mathcal{A}$  may only interact classically with honest parties. We write  $\mathcal{A}^{\mathcal{O}}$  to denote that  $\mathcal{A}$  has access to the oracle  $\mathcal{O}$ . We use  $\cdot$ , to represent required input to an algorithm, i.e.,  $\text{O}(\cdot, \cdot)$  denotes that the algorithm  $\text{O}$  takes two inputs.

Furthermore, we use  $y \leftarrow x$  to denote the assignment of a value  $x$  to a variable  $y$ . In security games, we use  $\llbracket \text{expression} \rrbracket$  to denote the boolean evaluation of *expression*. The special symbol  $\perp$  shall denote rejection or an error, usually output by an algorithm, in particular  $\perp \notin \{0, 1\}^*$ .

*Terminology.* In FIDO2, services are referred to as *relying parties (RP)*, to which users can authenticate via so-called *authenticators*. In this work, we view a user with their authenticator(s) as a singular actor and abstract away the client that is located between the authenticator and the relying party. This abstraction removes the analysis of the CTAP protocol but comes without loss of generality

as account recovery with ARKG is a WebAuthn extension, which is exactly the protocol executed between client and relying party.

Within ARKG, authenticators are split into two different classes: *backup* authenticators ( $BA$ ), which hold the long-term secrets denoted by  $(pk_{BA}, sk_{BA})$  and are used for account recovery, and *primary* authenticators ( $PA$ ) which derive (public) keys  $pk'$  and recovery information  $rec$  from the long-term key  $pk_{BA}$  and are used to authenticate the user to  $RPs$ .

## 2.1 ARKG Syntax

We now recall the notion of asynchronous remote key generation schemes introduced in [8] but slightly change notation to align it more with the intended purpose of account recovery in FIDO2.

We assume that the  $BA$  generates a long-term key pair  $(pk_{BA}, sk_{BA})$  via the algorithm  $KGen$ . Key pairs on the  $PA$  are denoted as  $(pk, sk)$ . They are generated together with recovery information  $rec$  via the algorithm  $DerivePK$  in such a way that allows the backup authenticator  $BA$  to recover the secret key with the help of  $sk_{BA}$  via the algorithm  $DeriveSK$ .

Both algorithms are linked through an algorithm  $Check$  to identify matching public and secret keys. Instead of calling the recovery information a *credential* denoted by  $cred$  as in [8] we call it *recovery information*, or  $rec$ , for short, resembling the externally stored session resumption data in TLS.

**Definition 1.** (ARKG). *A scheme for asynchronous remote key generation, or ARKG for short, consists of four algorithms ( $Setup, KGen, DerivePK, DeriveSK, Check$ ) such that*

*Setup takes as input the security parameter  $\lambda$  in unary and outputs the public parameters, i.e.,  $pp \leftarrow Setup(1^\lambda)$ .*

*KGen takes as input the public parameters  $pp$  and output a public/secret key pair  $(pk_{BA}, sk_{BA}) \leftarrow \$ KGen(pp)$  for the backup authenticator.*

*DerivePK takes as input the public parameters  $pp$ , a public key  $pk_{BA}$  and auxiliary information  $aux$ <sup>2</sup> and outputs a derived public key  $pk'$  and associated credential information  $rec$ , i.e.,  $(pk', rec) \leftarrow \$ DerivePK(pp, pk_{BA}, aux)$ .*

*DeriveSK takes as input the public parameters  $pp$ , a secret key  $sk_{BA}$  and recovery information  $rec$ . It outputs either a secret key  $sk'$ , i.e.,  $sk' \leftarrow DeriveSK(pp, sk_{BA}, rec)$ , or the dedicated symbol  $\perp$ , in case no valid  $sk'$  can be computed for  $pk'$  associated with  $rec$ .*

*Check takes as input a public-secret key pair  $(pk, sk)$  and returns 1 if  $(pk, sk)$  form a valid public/secret key pair, and 0 otherwise.*

*We say that an asynchronous remote key generation scheme  $ARKG = (Setup, KGen, DerivePK, DeriveSK, Check)$  is  $\epsilon$ -correct, if for all  $\lambda$  and  $pp \leftarrow Setup(1^\lambda)$*

<sup>2</sup> We assume that in the context of FIDO2 account recovery as treated in this paper,  $aux$  is a unique identifier  $rpId$  of the relying party for which the public key and credential are derived.

and  $(pk_{BA}, sk_{BA}) \leftarrow \$KGen(pp)$ , and auxiliary information  $aux$  we have that the probability of  $Check$  outputting 0 is bounded by  $\epsilon$ , i.e.,  $\Pr [Check(pk', sk') = 0] \leq \epsilon$ , for  $(pk', rec) \leftarrow \$DerivePK(pp, pk_{BA}, aux)$  and  $sk' \leftarrow DeriveSK(pp, sk_{BA}, rec)$ .

If the scheme is  $\epsilon$ -correct for  $\epsilon = 0$  then we say that the scheme is (perfectly) correct.

*Remark 1.* Frymann et al. include the algorithm  $Check(pk, sk)$  as part of their ARKG syntax, which is necessary to define correctness in the ARKG setting. In public-key cryptography, one can always leverage the randomness that went into key generation to implement such a check. That is, we define the secret key as the randomness used during key generation and, if required, reconstruct the actual secret key by re-running key generation. Then, one can easily check that the public key matches given the randomness as the secret key. Indeed, our generic construction follows this approach, such that we do not give a concrete instantiation of  $Check$ .

## 2.2 ARKG in the Context of FIDO2

As also mentioned in [8], the ARKG primitive itself may be applicable to use cases outside of account recovery for FIDO2. Most notably, privacy-preserving proxy signatures with unlinkable warrants can be generically constructed from ARKG [9]. This work focuses on the original motivation for the introduction of ARKG: FIDO2 account recovery.

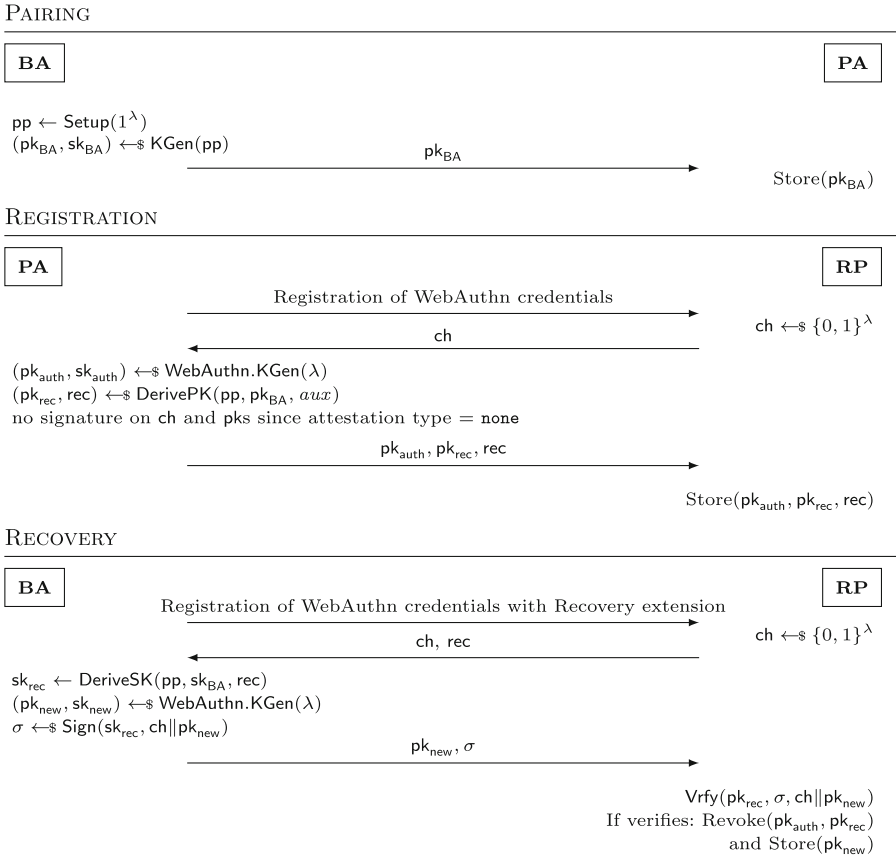
In Sect. 3, we will present our modified security notions for ARKG. To motivate the changes to the security definitions given in [8], we recap the basics of passwordless authentication via WebAuthn in FIDO2 [14] and how ARKG fits into this flow. Roughly speaking, the role of the ARKG primitive within the context of FIDO2 account recovery is two-fold:

On the *PA*: To create a signature key pair and recovery information from the *BA*'s long-term public key to register with relying parties such that no interaction with the *BA* is necessary to do so.

On the *BA*: To use the long-term secret and the recovery information from relying parties to derive a signing key to authenticate to the respective relying party and recover account access.

In more detail, ARKG has three phases: pairing, registration, and recovery. These phases are illustrated in Fig. 1, and we briefly describe them next.

**Pairing.** At the beginning, ARKG requires that the backup authenticator(s) be paired with a primary authenticator. We note that it is possible to pair any primary authenticator with multiple backup authenticators, and vice versa. Nonetheless, for ease of presentation, we focus on the case where a single *PA* is paired with a single *BA*. During the pairing process, the long-term public key  $pk_{BA}$  of the *BA* is transferred to the *PA* which stores it.



**Fig. 1.** Simplified illustration of how ARKG integrates into WebAuthn registration flows with default attestation type none.

**Registration.** At some point, the primary authenticator *PA* begins to register credentials with relying parties. This registration happens over a secure channel since the user has logged into the relying party via another authentication method, typically with a username and password, and has established a TLS connection to the server of the *RP*.

In a regular WebAuthn registration procedure, the relying party sends a challenge value to the authenticator, which then derives a key pair  $(pk_{auth}, sk_{auth})$  and sends  $pk_{auth}$  to the relying party. Depending on the chosen attestation type, the authenticator’s response may also include a signature on a message that contains (among other information) the challenge  $ch$  and the new public key  $pk_{auth}$ . This signature is created using the (highly non-unique) long-term secret key that is embedded in the authenticator at production time. Since no attestation is the proposed default, we chose to omit this signature from Fig. 1.

When the recovery extension is present, the *PA* will also derive a recovery public key  $\text{pk}_{\text{rec}}$  and recovery information  $\text{rec}$  from  $\text{pk}_{\text{BA}}$  via the ARKG algorithm `DerivePK`;  $(\text{pk}_{\text{rec}}, \text{rec})$  are then also transmitted to the *RP*.

**Recovery.** While the primary authenticator acts as the “standard” authenticator of the user when signing in to services, the backup authenticator comes into play should the user lose access to its *PA*. Until that point, the *BA* can be stored offline. Note, that the recovery process is a regular WebAuthn registration ceremony with the recovery extension. When the recovery is triggered by the *BA*, the relying party sends out a challenge  $\text{ch}$  to the authenticator along with recovery information  $\text{rec}$  for the user in question. The *BA* then uses its long-term secret  $\text{sk}_{\text{BA}}$  to recover the derived secret key  $\text{sk}_{\text{rec}}$  associated with  $\text{rec}$ . It then generates a new key pair  $(\text{pk}_{\text{new}}, \text{sk}_{\text{new}})$  to replace the lost  $(\text{pk}_{\text{auth}}, \text{sk}_{\text{auth}})$  and signs (among other information) the new public key  $\text{pk}_{\text{new}}$  and the challenge provided by the relying party. Finally, it sends the new public key and the signature to the relying party which then checks the signature wrt. its stored information. If the signature verifies, *RP* stores  $\text{pk}_{\text{new}}$  and should revoke the old stored credentials.

*Remark 2.* After the initial registration, the user can use their primary authenticator with the secret  $\text{sk}_{\text{auth}}$  to sign WebAuthn authentication challenges in a passwordless manner. ARKG is not involved in this phase, thus we did not include it in the Figure. As usual, this happens via a challenge-response protocol in which the relying party sends a challenge value to the user, and the user then signs the challenge with the secret key stored on the authenticator. The *RP* then verifies the signature with respect to the public key it had received during registration. If the signature is valid, the user is authenticated and logged onto the service.

### 3 Security of ARKG Schemes

When first introducing ARKG and in later works, Frymann et al. [7, 8, 10] described security in terms of an adversary’s inability to recover a derived secret key in various adversarial settings (*honest/malicious*, *weak/strong*) and the unlinkability of derived public keys. The former should guarantee that an adversary is not able to successfully complete the account recovery process without access to the secret key stored on the backup authenticator, whereas public-key unlinkability shall ensure that users cannot be tracked across services via their registered public-key credentials. Unfortunately, neither of the previously proposed notions adequately provides these guarantees in the FIDO2 setting.

The key security notion in [8] demands that in order to break the scheme, an adversary must be able to recover the *entire* secret key. However, in the context of FIDO2, recovery is broken if an adversary can successfully convince a relying party that they are authorized to register new credentials following a recovery. For this, the adversary does not need knowledge of the full secret key. Thus, we switch to a notion based on the adversary’s (in)ability to successfully *authenticate*.

With regards to public-key unlinkability, we note that the definition in [8], which states that derived keys are indistinguishable from randomly sampled keys, does not take the adversary’s actual view during the execution of the protocol into account. This omission gives a false sense of security: One can have ARKG schemes that provide public-key unlinkability wrt. Frymann et al.’s definition that trivially link derived public keys when employed in the envisioned setting.

In the following, we provide the security notions of *authentication security* and *unlinkability* that fix the just mentioned shortcomings. Before we formally define these security properties for ARKG schemes, we first state our basic assumptions on the adversary’s power and capabilities.

The formal definitions from [8] can be found in Appendix A.

### 3.1 Adversarial Model

Recall that we assume a quantum polynomial-time (QPT) adversary since our ARKG construction aims to provide post-quantum security, interacting classically with the honest parties, i.e., it may not query any oracles in superposition.

We note that it is generally assumed that authenticators are tamperproof, i.e., they do not leak information on the secret keys stored on them, even if they are in the adversary’s possession. This assumption was also made for the FIDO2 analysis by Barbosa et al. [1] and is intuitively also reasonable in our setting where we assume the primary authenticator has been lost, i.e., might be in the hands of the adversary. If (primary) authenticators leaked secret keys, the adversary could immediately log into services and reset credentials such that account recovery would not be possible anymore.

Nevertheless, we provide the adversary with oracles that leak the derived secret keys to achieve stronger notions of security by default, analogous to the strong version in [7, 8, 10].

We assume that the initial pairing between the *BA* and the *PA(s)* is in a trusted setting such that an adversary is not able to inject its own long-term public key into the user’s primary authenticator. This is a reasonable assumption since this pairing only happens once, is of short duration, and is executed locally at the user with no information going over public network channels.

Backup authenticators are typically offline and should only come online during account recovery. Since we cannot rule out that an adversary intercepts the user’s account recovery attempts, we do, nevertheless, grant the adversary access to a signing oracle where the *BA*’s long-term secrets are employed.

We assume that WebAuthn registrations (with extensions) are secure against active adversaries (cf. [3]). In the context of ARKG, this is especially important during registration, where the derived public keys and recovery information are transmitted from the *PA* to the relying party. If the adversary were able to inject its own account recovery credentials here, all is lost. This assumption is in line with the security model for FIDO2.

Typically, upon registration of FIDO2 credentials, a user has previously logged in to the service using other means of authentication, e.g., with username and password, and has established an authenticated connection [1]. Thus,



we assume that the adversary remains passive during the registration of credentials with a relying party. During the account recovery process, however, the user is not authenticated to the relying party and no secure channel exists. Thus, we allow the adversary to actively interfere, i.e., it may drop, modify, or inject messages.

We remind the reader of the possibility of pairing any primary authenticator with multiple backup authenticators, and vice versa. For the former case, all interactions are now carried out for each of the registered backup authenticators in parallel. This introduces a minor side channel by revealing the number of registered backup authenticators to relying parties. For the latter case, no change in the protocol is required and the security remains unaffected.

As usual for reliable authentication, we assume that public keys are globally unique. This can be accomplished by including the relying party's identity  $rp_{id}$  and a unique user identifier (or pseudonym)  $uid$  in the public key.

### 3.2 Authentication Security

Viewed merely from the cryptographic primitive level, the main functionality of ARKG schemes is to derive public-secret key pairs  $(pk', sk')$  along with additional recovery information  $rec$  from a long-term public key  $pk_{BA}$  such that  $sk'$  can only be recovered with knowledge of the long-term secret  $sk_{BA}$ . Frymann et al. [8] thus describe the main security property of ARKG schemes as one where an adversary may not be able to derive valid public-secret key pairs (and recovery information) without knowledge of the long-term secret key.

As elaborated in Sect. 2.2, ARKG schemes were originally introduced to support account recovery in case of primary authenticator loss in FIDO2 authentication procedures, i.e., in a challenge-response-based protocol using digital signatures. Viewed in this context, the main security goal of ARKG schemes should be the adversary's inability to create a valid response, i.e., a valid signature on a given challenge value (and new public key credential) during an account recovery procedure. Since our main contribution is a generic post-quantum secure ARKG construction for account recovery in FIDO2 authentications, we opt to define security in the latter sense as follows.

We discuss the differences between this notion, and the one given by Frymann et al. in more detail in Appendix A.

*Game Description.* The formal description of the authentication game  $\text{Exp}_{\text{ARKG}}^{\text{auth}}(\mathcal{A})$  is given in Fig. 2. The adversary  $\mathcal{A}$  gets as input the public parameters  $pp$  and the long-term public key  $pk_{BA}$  from the backup authenticator  $BA$ .  $\mathcal{A}$  then has access to the oracles  $\text{DERIVEPK}$ ,  $\text{CHALL-AUTH}$ ,  $\text{SIGN}$ , and  $\text{LEAKSK}$ .

The oracle  $\text{DERIVEPK}$  takes as input auxiliary data  $aux$  and derives a public key  $pk'$  and recovery information  $rec$  for the relying party specified in  $aux$  from the long-term public key  $pk_{BA}$ . This simulates the honest generation of derived public keys and recovery information on the primary authenticator  $PA$  when registering with relying parties specified in the auxiliary data  $aux$ .

$\text{Exp}_{\text{ARKG}}^{\text{auth}}(\mathcal{A})$ :

```

1  $\text{pp} \leftarrow \text{Setup}(1^\lambda)$ 
2  $\mathcal{L}_{\text{keys}}, \mathcal{L}_{\text{ch}}, \mathcal{L}_{\text{sk}'}, \mathcal{L}_\sigma \leftarrow \emptyset$ ;
3  $(\text{pk}_{\text{BA}}, \text{sk}_{\text{BA}}) \leftarrow \text{KGen}(\text{pp})$ 
4  $(\text{pk}^*, \text{rec}^*, \text{aux}^*, m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{DERIVEPK}, \text{CHALL-AUTH}, \text{SIGN}, \text{LEAKSK}}(\text{pp}, \text{pk}_{\text{BA}})$ 
5 return  $\llbracket (\text{pk}^*, \text{rec}^*, \text{aux}^*) \in \mathcal{L}_{\text{keys}} \wedge \exists (\text{ch}, \text{aux}^*) \in \mathcal{L}_{\text{ch}} : \text{prefix}(m^*) = \text{ch} \wedge \text{Vrfy}(\text{pk}^*, \sigma^*, m^*) \wedge (\text{rec}^*, m^*) \notin \mathcal{L}_\sigma \wedge \text{rec}^* \notin \mathcal{L}_{\text{sk}'} \rrbracket$ 
    
```

$\text{DERIVEPK}(\text{pp}, \text{pk}_{\text{BA}}, \cdot)$  on input  $\text{aux}$ :

```

6  $(\text{pk}', \text{rec}) \leftarrow \text{DerivePK}(\text{pp}, \text{pk}_{\text{BA}}, \text{aux})$ 
7  $\mathcal{L}_{\text{keys}} \leftarrow \mathcal{L}_{\text{keys}} \cup \{(\text{pk}', \text{rec}, \text{aux})\}$ 
8 return  $(\text{pk}', \text{rec})$ 
    
```

$\text{CHALL-AUTH}(\cdot)$  on input  $\text{aux}$ :

```

9  $\text{ch} \leftarrow \{0, 1\}^\lambda$ 
10  $\mathcal{L}_{\text{ch}} \leftarrow \mathcal{L}_{\text{ch}} \cup \{(\text{ch}, \text{aux})\}$ 
11 return  $\text{ch}$ 
    
```

$\text{SIGN}(\cdot, \cdot)$  on input  $(\text{rec}, m)$ :

```

12  $\text{sk}' \leftarrow \text{DeriveSK}(\text{pp}, \text{sk}_{\text{BA}}, \text{rec})$ 
13 if  $\text{sk}' = \perp$ : abort
14  $\sigma \leftarrow \text{Sign}(\text{sk}', m)$ 
15  $\mathcal{L}_\sigma \leftarrow \mathcal{L}_\sigma \cup \{(\text{rec}, m)\}$ 
16 return  $\sigma$ 
    
```

$\text{LEAKSK}(\cdot)$  on input  $\text{rec}$ :

```

17  $\text{sk}' \leftarrow \text{DeriveSK}(\text{pp}, \text{sk}_{\text{BA}}, \text{rec})$ 
18  $\mathcal{L}_{\text{sk}'} \leftarrow \mathcal{L}_{\text{sk}'} \cup \{\text{rec}\}$ 
19 return  $\text{sk}'$ 
    
```

**Fig. 2.** Our security definition for authentication security of ARKG schemes.

As they are by default not authenticated, account recovery processes may be triggered by the adversary. Thus,  $\mathcal{A}$  gets access to the challenge oracle  $\text{CHALL-AUTH}$ , which takes as input auxiliary data  $\text{aux}$  and outputs a uniformly random challenge value  $\text{ch}$ . This challenge value corresponds to the challenges sent out by the relying parties that are specified via  $\text{aux}$  in the account recovery process. The adversary eventually has to create a valid signature on a message containing one of these challenges, more specifically on a message  $m$  that starts with a challenge value and has not been queried to  $\text{SIGN}$  with respect to the secret key.

When it receives some recovery information  $\text{rec}$ , the backup authenticator  $BA$  has no means to distinguish between credential information that had been honestly generated by a primary authenticator and recovery information that the adversary sends to it. The  $BA$  will simply use its long-term secret  $\text{sk}_{\text{BA}}$  to derive the secret key  $\text{sk}'$  and sign the response with it. Thus, we grant  $\mathcal{A}$  access to an oracle  $\text{SIGN}$  which takes as input recovery information  $\text{rec}$  and a message  $m$ . The oracle then tries to derive a secret key  $\text{sk}'$  and signs the provided message  $m$ , returning the signature  $\sigma$ . If the secret key derivation fails, the oracle simply aborts.

The  $\text{LEAKSK}$  oracle models the leakage of derived secret keys. The adversary may provide recovery information  $\text{rec}$  and the oracle will return the output of  $\text{DeriveSK}$ , which is either  $\perp$  if the derivation failed or the derived secret key  $\text{sk}'$ .

The adversary outputs  $(\text{pk}^*, \text{rec}^*, \text{aux}^*, m^*, \sigma^*)$  and wins the game  $\text{Exp}_{\text{ARKG}}^{\text{auth}}(\mathcal{A})$ , if it is able to produce a valid signature on a message containing the challenge posed by a relying party. The valid signature must fulfill the following requirements:

- $(pk^*, rec^*)$  was honestly generated for the relying party specified in  $aux^*$ ,
- there exists an honestly generated challenge  $ch$  for  $aux^*$  such that  $ch$  is a prefix of  $m^*$ ,
- $\sigma^*$  is a valid signature on  $m^*$  with respect to  $pk^*$ ,
- the adversary has not received a signature on  $m^*$  with respect to the secret key associated with  $rec^*$ , and
- the secret key associated with  $rec^*$  has not been given to the adversary.

**Definition 2.** Let  $ARKG = (\text{Setup}, \text{KGen}, \text{DerivePK}, \text{DeriveSK})$  be an *async. remote key generation scheme*. We say that  $ARKG$  is *AUTH-secure*, if for every *QPT adversary*  $\mathcal{A}$  the advantage in winning the game  $\text{Exp}_{ARKG}^{\text{auth}}(\mathcal{A})$  described in Fig. 2, defined as

$$\text{Adv}_{ARKG, \mathcal{A}}^{\text{auth}}(\lambda) := \left| \Pr [\text{Exp}_{ARKG}^{\text{auth}}(\mathcal{A}) = 1] \right|$$

is negligible in the security parameter  $\lambda$ .

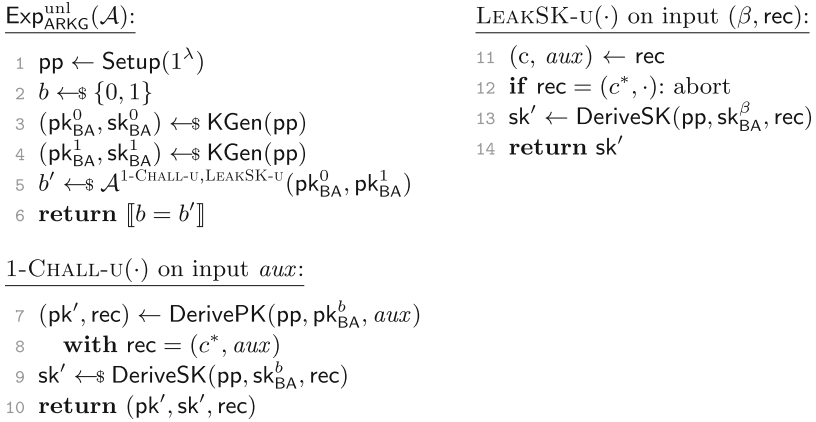
*Remark 3.* Note that a weaker notion of authentication security, where  $\mathcal{A}$  does not have access to the  $\text{LEAKSK}$  oracle to learn other derived keys, could be defined. However, at least in our instantiation from  $\text{KEMs}$ , we gain nothing from this modification as both notions require the same assumptions on the primitives.

### 3.3 Unlinkability

Unlinkability aims to fulfill a requirement in the  $\text{WebAuthn}$  standard [14] which recommends authenticators to ensure that the credential IDs and credential public keys of different public-key credentials cannot be correlated as belonging to the same user. We note that this is a non-normative requirement, i.e.,  $\text{WebAuthn}$  implementations that do not provide this unlinkability are still considered as conforming to the standard. As mentioned, we deviate from Frymann et al.’s definition of public-key unlinkability, which was based on the adversary’s inability to distinguish derived from randomly sampled key pairs.

In essence, their definition allows to prove  $ARKG$  schemes as public-key unlinkable although they trivially link public-key credentials when employed in account recovery.

In our definition, two long-term key pairs  $(pk_{BA}^0, sk_{BA}^0)$  and  $(pk_{BA}^1, sk_{BA}^1)$  are generated and the public keys are given to the adversary. A bit  $b \leftarrow \{0, 1\}$  is sampled uniformly at random. The adversary may once query auxiliary information of its choice to the oracle  $1\text{-CHALL-U}$ . The oracle then derives a public key  $pk'$  and credential information  $rec$  either from  $pk_{BA}^0$  (if  $b = 0$ ), or  $pk_{BA}^1$  (if  $b = 1$ ). It then derives the corresponding secret key  $sk'$  and outputs  $(pk', sk', rec)$  as the challenge to the adversary. Note that with a standard hybrid argument, one may lift this definition to a setting with multiple challenges. Additionally, the adversary may learn derived secret keys  $sk'$  for credential information of its choice, where it can also specify via a bit  $\beta$  which of the long-term secrets  $sk_{BA}^\beta$



**Fig. 3.** Our security definition for unlinkability of ARKG schemes.

shall be used in the oracle’s internal `DeriveSK` call. Note that if secret key derivation fails, `DeriveSK` outputs  $\perp$  and this is then returned to the adversary as  $\text{sk}'$ . Of course, the adversary may not query its challenge ciphertext to the oracle `LEAKSK-U`, even if the auxiliary information in the recovery credential has been modified. This does not impose any undue limitation, because during an honest execution, the auxiliary information is the unique identifier of the relying party and thus remains unchanged. In the end,  $\mathcal{A}$  will output a bit  $b'$ , guessing whether the challenge was derived from  $\text{pk}_{\text{BA}}^0$  or  $\text{pk}_{\text{BA}}^1$  and wins if correct. More formally,

**Definition 3.** *As before, let  $\text{ARKG} = (\text{Setup}, \text{KGen}, \text{DerivePK}, \text{DeriveSK})$  be an asynchronous remote key generation scheme. We say that ARKG provides unlinkability, or is UNL -secure, for short, if for every QPT adversary  $\mathcal{A}$ , the advantage in winning the game  $\text{Exp}_{\text{ARKG}}^{\text{unl}}(\mathcal{A})$  described in Fig. 3, defined as*

$$\text{Adv}_{\text{ARKG}, \mathcal{A}}^{\text{unl}}(\lambda) := \left| \Pr [\text{Exp}_{\text{ARKG}}^{\text{unl}}(\mathcal{A}) = 1] - \frac{1}{2} \right|$$

is negligible in the security parameter  $\lambda$ .

Recall that [8] in their (public-key) unlinkability game asks to distinguish genuinely generated public keys against independently sampled ones. This requires defining a distribution of public keys. We have opted here for the common left-or-right notion. In principle, we could also cover such real-or-random scenarios. Looking ahead to a post-quantum instantiation, the post-quantum ML-KEM Kyber, currently being standardized as FIPS 203, achieves this notion. The reason is that Kyber provides *strong pseudorandomness* under CCA [23], as shown in [18].

## 4 Post-quantum Asynchronous Remote Key Generation

This section will introduce our instantiation for PQ-ARKG, built from generic primitives, and provide security proofs in the setting discussed in Sect. 3. Choosing generic primitives for the instantiation allows us to provide a general security proof independent of the actual instantiation of the primitives. The result ensures ARKG is secure within the specified scenario, as long as the underlying primitives achieve the respective security properties.

### 4.1 The PQ-ARKG Scheme

In Fig. 4 we provide the generic instantiation for all algorithms required for an ARKG scheme. The key building blocks of the proposed ARKG instantiation are key encapsulation mechanisms, digital signatures and key derivation functions, all of which allow for multiple concrete instantiations believed to be resistant to a QPT attacker with high confidence [15, 17, 19, 20]. Formal definitions of the building blocks used in this chapter can be found in Appendix B. Conceptually, the interactions that make up a full ARKG protocol execution work as follows: During pairing, the *BA* generates a KEM key pair, denoted as  $(pk_{BA}, sk_{BA})$ , and transfers  $pk_{BA}$  to the *PA*.

<u>Setup (<math>1^\lambda</math>):</u> 1 <b>return</b> $pp = (\text{KEM}, \text{KDF}, \text{Sig})$	<u>KGen(pp):</u> 1 $(pk_{BA}, sk_{BA}) \leftarrow_{\$} \text{KEM.KGen}(pp)$
<u>DerivePK(pp, <math>pk_{BA}</math>, <math>aux</math>):</u> 1 $(c, K) \leftarrow_{\$} \text{KEM.Encaps}(pk_{BA})$ 2 $r \leftarrow \text{KDF}(K, aux)$ 3 $(pk', sk') \leftarrow \text{Sig.KGen}(pp; r)$ 4 <b>rec</b> $\leftarrow (c, aux)$	<u>DeriveSK(pp, <math>sk_{BA}</math>, <math>rec</math>):</u> 1 $(c, aux) \leftarrow \text{rec}$ 2 $K \leftarrow \text{KEM.Decaps}(sk_{BA}, c)$ 3 $r \leftarrow \text{KDF}(K, aux)$ 4 $(pk', sk') \leftarrow \text{Sig.KGen}(pp; r)$

**Fig. 4.** Our PQ-ARKG instantiation from KEMs, Signatures, and KDFs

During registration, which is exclusively done by the *PA*, an encapsulation operation is performed under  $pk_{BA}$  to obtain a random key, which is then input to a KDF which outputs a random seed in the desired format. This seed is then used to deterministically generate a new signature key pair  $(pk', sk')$ . The ciphertext resulting from the encapsulation operation is sent to the relying party for safekeeping along with the newly derived public key  $pk'$ .

During recovery, the *BA* retrieves the ciphertext from the *RP* and performs a decapsulation operation to obtain the key used as input to the KDF. By executing the PRF, it obtains the seed used for the key generation. This allows *BA* to regenerate the original signature key pair, which critically includes the secret key  $sk'$ . As a result, *BA* now has access to the same signing key pair as *PA* had during the registration, without any direct communication from *PA* to *BA*.

In short, we use KEM ciphertexts stored at the relying parties to securely relay seeds for the creation of recovery credentials between  $PA$  and  $BA$ .

A minor difference between the instantiation proposed in [8] and in this work is the fact that the primary authenticator temporarily has access to the full recovery key pair  $(pk', sk')$ . Nonetheless, the secret key material is immediately discarded by the primary authenticator after the generation of  $pk'$ . This does not pose a security risk, as the primary authenticator is also in possession of the primary credentials used during regular FIDO2 sessions. Consequently, an attacker with access to the primary authenticator's internal secrets could authenticate himself using a regular FIDO2 interaction while completely disregarding the recovery extension. The fact that recovery credentials are generated by the primary authenticator but only ever used by the backup authenticator therefore also holds for our instantiation.

## 4.2 Security Analysis

Our instantiation of ARKG achieves both authentication security, as well as unlinkability, as reflected in the following theorems. The standard definitions of security for the employed cryptographic primitives can be found in Appendix B.

**Authentication Security.** We first show authentication security.

**Theorem 1.** *Let ARKG be the generic instantiation of ARKG as given in Fig. 4, KEM be an IND-CCA secure and  $\epsilon$ -correct KEM scheme, Sig be an EUF-CMA secure signature scheme and KDF a secure key derivation function modeled as a PRF. Then ARKG provides  $\epsilon$ -correctness and authentication security as defined in Definition 2. More precisely, for any QPT adversary  $\mathcal{A}$  against AUTH, there exist QPT algorithms  $\mathcal{B}_1, \mathcal{B}_2$ , and  $\mathcal{B}_3$  with approximately the same running time as  $\mathcal{A}$  such that*

$$\text{Adv}_{\text{ARKG}, \mathcal{A}}^{\text{auth}}(\lambda) \leq q \cdot \left( \epsilon + \text{Adv}_{\text{KEM}, \mathcal{B}_1}^{\text{ind-cca}}(\lambda) + \text{Adv}_{\text{KDF}, \mathcal{B}_2}^{\text{prf}}(\lambda) + \text{Adv}_{\text{Sig}, \mathcal{B}_3}^{\text{euf-cma}}(\lambda) \right)$$

where  $q$  is the maximum number of calls to the DERIVEPK oracle.

*Proof.* Correctness with parameter  $\epsilon$  for ARKG directly follows from  $\epsilon$ -correctness of KEM: If one is able to decapsulate the right key, then one can also derive the same key pair. We will prove the authentication property of Theorem 1 using game hopping. We denote by  $\text{Adv}_{\text{ARKG}, \mathcal{A}}^{\text{game}_i}(\lambda)$  the advantage of the adversary in the corresponding game.

**Game<sub>1</sub>( $\lambda$ ):** The original AUTH security game  $\text{Exp}_{\text{ARKG}}^{\text{auth}}(\mathcal{A})$ .

**Game<sub>2</sub>( $\lambda$ ):** In this game we assume that KEM decapsulation for honestly generated public keys and ciphertexts never fails. This is always the case, except for a negligible failure probability  $\epsilon$  for each of the at most  $q$  generated keys, given by Definition 1. Thus, we get the bound

$$\text{Adv}_{\text{ARKG}, \mathcal{A}}^{\text{game}_1}(\lambda) \leq q \cdot \epsilon + \text{Adv}_{\text{ARKG}, \mathcal{A}}^{\text{game}_2}(\lambda).$$

**Game<sub>3</sub>(λ):** In this game we guess for which call of DERIVEPK the adversary will output the forgery  $(pk^*, rec^*, aux^*, m^*, \sigma^*)$  for the key  $pk^*$  output by DERIVEPK. Note that, by definition, the adversary must succeed for one of the keys in  $\mathcal{L}_{keys}$ . We denote the number of oracle calls of  $\mathcal{A}$  to DERIVEPK with  $q$ . Consequently, the correct oracle call is guessed with a probability of  $1/q$  and hence it follows that

$$Adv_{ARKG, \mathcal{A}}^{game_3}(\lambda) \leq q \cdot Adv_{ARKG, \mathcal{A}}^{game_3}(\lambda).$$

**Game<sub>4</sub>(λ):** In this game we modify the behavior of the DerivePK algorithm for the execution guessed during the previous game: The input to the KDF, which previously was a KEM ciphertext, is replaced with a random value. This substitution takes place in line 2 of the DerivePK algorithm (cf. Figure 4).

We show that any efficient adversary  $\mathcal{A}$ , which can distinguish between **game<sub>3</sub>** and **game<sub>4</sub>** implies the existence of an efficient adversary  $\mathcal{B}_1$  against the IND-CCA security of KEM.  $\mathcal{B}_1$  receives the KEM challenge  $(pk^*, k^*, c^*)$  and initializes  $Exp_{ARKG}^{auth}(\mathcal{A})$  with  $pk^*$  as  $pk_{BA}$ .

During the execution of DerivePK that has been guessed in **game<sub>3</sub>**, algorithm  $\mathcal{B}_1$  modifies the behavior of the algorithm by plugging in its own challenge: In line 1 of the DerivePK algorithm,  $pk^*$  is used for encapsulation; in line 2  $k^*$  is used as input to the KDF. The ciphertext, which is output as a component of  $rec$ , is replaced with the ciphertext  $c^*$ .

To simulate the SIGN Oracle, the reduction keeps a list of derived secret keys, which are also generated as part of the DerivePK algorithm, but discarded during normal operation. As we are only retrieving stored keys, we implicitly eliminate all decryption failures, which is already captured by the transition to **game<sub>2</sub>**. Queries to the SIGN oracle for inputs that have not been generated by the DerivePK algorithm can be answered using the Decaps oracle provided by the IND-CCA challenger. Due to the guess in **game<sub>3</sub>**, the challenge key and ciphertext is always embedded in an output of the DerivePK oracle, and therefore such a query to the Decaps oracle does not coincide with the challenge ciphertext, which subsequently means that the game’s own Decaps oracle always answers.

To simulate LEAKSK our reduction  $\mathcal{B}_1$  needs to answer queries  $rec$  to LEAKSK without knowing the decryption key  $sk_{BA}$  of the KEM. But since the adversary can only win if the forgery attempt  $rec^*$  does not lie in  $\mathcal{L}_{sk'}$ , reduction  $\mathcal{B}_1$  can use its own decryption oracle of the IND-CCA KEM to answer these different requests.

CHALL-AUTH can be trivially simulated, as it has no secret inputs.

Finally,  $\mathcal{A}$  terminates and outputs a guess  $b$ , which the reduction  $\mathcal{B}_1$  outputs as its own answer to the KEM challenger. Clearly,  $\mathcal{B}_1$  perfectly simulates **game<sub>3</sub>** when the KEM challenge is real and **game<sub>4</sub>** when the KEM challenge is random. Consequently, we obtain the following bound:

$$Adv_{ARKG, \mathcal{A}}^{game_3}(\lambda) \leq Adv_{KEM, \mathcal{B}_1}^{ind-cca}(\lambda) + Adv_{ARKG, \mathcal{A}}^{game_4}(\lambda).$$

**Game<sub>5</sub>(λ):** In this game the execution of the `DerivePK` algorithm is further modified. The variable  $r$ , which was previously assigned the output of a KDF, is now sampled uniformly at random.

Any efficient adversary  $\mathcal{A}$ , able to distinguish `game3` and `game4` can be used to construct an efficient adversary  $\mathcal{B}_2$  against the security of the underlying KDF, whose security we model as a PRF. The construction works similarly as in the previous game hop.  $\mathcal{B}_2$  initializes  $\text{Exp}_{\text{ARKG}}^{\text{auth}}(\mathcal{A})$  for  $\mathcal{A}$  as specified, but modifies the behavior of the KDF used as part of the `DerivePK` algorithm in line 2 (cf. Figure 4). Instead of directly invoking the key derivation function,  $\mathcal{B}_2$  forwards the input to the PRF oracle provided by the PRF challenger.

The simulation of the other oracles works identically as in the previous hop. Finally,  $\mathcal{A}$  terminates and outputs a bit  $b$ , to indicate whether it is playing against `game3` or `game4`.  $\mathcal{B}_2$  forwards this as its own output to the PRF challenger.

Clearly,  $\mathcal{B}_2$  perfectly simulates `game3` if the oracle is an actual KDF, and `game4` if the oracle is a random function. Thus, we get the following advantage:

$$\text{Adv}_{\text{ARKG}, \mathcal{A}}^{\text{game}_4}(\lambda) \leq \text{Adv}_{\text{KDF}, \mathcal{B}_2}^{\text{prf}}(\lambda) + \text{Adv}_{\text{ARKG}, \mathcal{A}}^{\text{game}_5}(\lambda).$$

Now we bound the last term on the right hand side. For this we can construct a reduction  $\mathcal{B}_3$ , which uses an efficient adversary  $\mathcal{A}$  against `game4` as a subroutine and can efficiently win against any EUF-CMA challenger with non-negligible probability. This allows us to bound the advantage of any QPT adversary against `game4` by the EUF-CMA security of the underlying signature scheme.

The reduction  $\mathcal{B}_3$  receives a challenge public key  $\text{pk}^*$  and a signing oracle `SIGN` from the EUF-CMA challenger. It then initializes the game `game4` as specified, in particular it holds the backup authenticator's key pair  $(\text{pk}_{\text{BA}}, \text{sk}_{\text{BA}})$ . During the query guessed in the first game hop, it replaces the public key output by `DerivePK` with the challenge public key  $\text{pk}^* = \text{pk}^*$ . Note that this also means that this choice also determines the recovery information  $\text{rec}^*$ . Replacing the public key by  $\text{pk}^*$  is possible, as in `game4` the output of `DerivePK` is completely independent of both the key derivation function and the initial public key  $\text{pk}_{\text{BA}}$ .

Queries by  $\mathcal{A}$  to the `Sign` Oracle of the AUTH game for the value  $\text{rec}^*$  can be forwarded to the outer `Sign` Oracle of the EUF-CMA game by the reduction  $\mathcal{B}_3$ . Since `DeriveSK` is deterministic, the signature oracle in the attack would recover exactly *the* secret key to  $\text{pk}^*$ , such that using the external signing oracle is valid. Note that signature queries for any other  $\text{rec}$  value can be answered with the help of  $\text{sk}_{\text{BA}}$ , first recovering the derived key and then signing the input message  $m$ .

Ultimately, the inner adversary  $\mathcal{A}$  terminates and outputs values  $(\text{pk}^*, \text{rec}^*, \text{aux}^*, m^*, \sigma^*)$ , where  $\sigma^*$  is a valid signature under the challenge public key  $\text{pk}^*$  and an arbitrary message  $m^*$ . The reduction can then output the message-signature pair  $(m^*, \sigma^*)$  as its forgery. Per construction, this constitutes a valid forgery: The only queries forwarded to  $\mathcal{B}_3$ 's external signing oracle are the ones for  $\text{rec}^*$ . Since the adversary  $\mathcal{A}$  can only win if  $(\text{rec}^*, m^*)$  is not in the list of signed pairs  $\mathcal{L}_\sigma$ , it follows that  $m^*$  must not have been signed before in  $\mathcal{B}_3$ 's attack.



Consequently, the success probabilities of  $\mathcal{A}$  and  $\mathcal{B}_3$  are equal. Thus we can conclude that

$$\text{Adv}_{\text{ARKG},\mathcal{A}}^{\text{game}_5}(\lambda) \leq \text{Adv}_{\text{Sig},\mathcal{A}}^{\text{euf-cma}}(\lambda).$$

To conclude the proof, we sum up the advantages:

$$\text{Adv}_{\text{ARKG},\mathcal{A}}^{\text{auth}}(\lambda) \leq \epsilon + q \cdot \left( \text{Adv}_{\text{KEM},\mathcal{B}_1}^{\text{ind-cca}}(\lambda) + \text{Adv}_{\text{KDF},\mathcal{B}_2}^{\text{prf}}(\lambda) + \text{Adv}_{\text{Sig},\mathcal{B}_3}^{\text{euf-cma}}(\lambda) \right).$$

Note that in the proof we have not used the requirement that the forgery needs to be for a random challenge and for the right format. The reason is that we presume existential unforgeability of the signature scheme, such that even forgeries for arbitrary messages should be infeasible. For practical purposes we would only require the relaxed unforgeability notion but do not explore this here further.

**Unlinkability.** We next discuss the unlinkability of our scheme. This follows from the fact that the underlying KEM scheme is anonymous [2].

**Theorem 2.** *Let ARKG be the instantiation of ARKG as shown in Fig. 4 and KEM be an ANON-CCA secure KEM scheme. Then ARKG provides unlinkability security as described in Definition 3. More precisely, for any QPT adversary  $\mathcal{A}$  against UNL there exists a QPT algorithm  $\mathcal{B}$  with approximately the same running time as  $\mathcal{A}$ , such that*

$$\text{Adv}_{\text{ARKG},\mathcal{A}}^{\text{unl}}(\lambda) \leq \text{Adv}_{\text{KEM},\mathcal{B}}^{\text{anon-cca}}(\lambda).$$

*Proof.* We prove Theorem 2 using a direct reduction to the ANON-CCA security of the underlying KEM. Let  $\mathcal{A}$  be a QPT adversary against unlinkability. We use  $\mathcal{A}$  to construct an efficient reduction,  $\mathcal{B}$ , that uses  $\mathcal{A}$  as a subroutine to win against ANON-CCA with non-negligible probability.

First,  $\mathcal{B}$  receives the challenge set  $(\text{pk}_0, \text{pk}_1, c^*, k^*)$  as per the ANON-CCA security definition (cf. Figure 8). Then,  $\mathcal{B}$  forwards  $(\text{pk}_0, \text{pk}_1)$  to the inner adversary  $\mathcal{A}$  as  $(\text{pk}_{\text{BA}}^0, \text{pk}_{\text{BA}}^1)$ . Next,  $\mathcal{A}$  outputs  $aux$  to query the 1-CHALL-U oracle. The reduction simulates the behavior of 1-CHALL-U as follows: During the execution of the algorithm DerivePK (cf. Figure 4), the challenge key  $k^*$  is used as input to the KDF in combination with  $aux$  provided by the inner adversary  $\mathcal{A}$ . The KDF output is then used as input to the key generation algorithm. Lastly, it creates  $\text{rec}$  as  $\text{rec} \leftarrow (c^*, aux)$ . Then it returns  $(\text{pk}', \text{sk}', \text{rec})$  to the inner adversary. Queries to the LEAKSK-U oracle can be answered by  $\mathcal{B}$  with the help of its own decapsulation oracle provided by the ANON-CCA challenger; any query including about the challenge value  $c^*$  is immediately rejected.

Finally,  $\mathcal{A}$  outputs a bit  $b$ , which the reduction forwards as its guess to the ANON-CCA challenger. Depending on the bit  $b$  of the ANON-CCA game, this perfectly simulates either the case where the challenge bit of UNL is sampled as 0 or 1.

We have constructed  $\mathcal{B}$  in such a way, that it is efficient and perfectly simulates the UNL game for  $\mathcal{A}$  and its view depends only on the random bit  $b$  chosen

by the challenger. Consequently, the success probability of the reduction is equal to that of the inner adversary, which yields the following result:

$$\text{Adv}_{\text{ARKG},\mathcal{A}}^{\text{unl}}(\lambda) \leq \text{Adv}_{\text{KEM},\mathcal{B}}^{\text{anon-cca}}(\lambda).$$

### 4.3 Overhead and Instantiation

Implementing ARKG requires relatively low additional computations and storage at both the relying party and the authenticator itself, with the overhead dependent on the instantiation of the underlying primitives. Crucially, during the most frequent operation, namely authentication, no additional computations are necessary. For each registration of an authenticator at a relying party, only a single additional KEM key generation and encapsulation are performed. Similarly, the initial pairing (only done once) and account recovery require a KEM key generation and one single other operation (KEM decapsulation and signing, respectively). In terms of storage, the authenticator needs to store the backup authenticator’s public key and the relying party needs to store the public key of the recovery credential and the recovery information  $\text{rec}$ .

Our solution can be instantiated with any suitable primitive that satisfies the security requirements stated in the theorems. For example, SPHINCS+ could be a viable signature choice due to its small key sizes, which would be beneficial for the storage overhead at the relying parties, however, the recovery would take longer than with other options. We leave the ideal tradeoff between storage and computational costs as an open question for future work.

**Acknowledgements.** We thank Varun Maram for pointing out flaws in the security proofs of a previous version of this work as well as the anonymous reviewers for their valuable comments. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - SFB 1119-236615297 and the German Federal Ministry of Education and Research (BMBF) under reference 16KISQ074.

## A Comparison to the Different Security Definitions

In the following, we review the security definitions of ARKG schemes as proposed by Frymann et al. [7,8,10] and compare them to our proposed notions.

### A.1 Key Security

Intuitively, key security formalizes the confidentiality of the derived secret keys and assures that no party can learn the key without knowledge of the BAs secret key. The notions by Frymann et al. [8] for *key security* are based on the adversary’s inability to output an entire valid secret key  $\text{sk}^*$  needed for account recovery. We have depicted the strong and weak game description for SK-security of [8] in the *honest* setting in Fig. 5.

<sup>3</sup> We note that the pseudocode descriptions of  $\mathcal{O}_{\text{pk}'}$  and  $\mathcal{O}_{\text{sk}'}$  have not been given before and thus correspond merely to our interpretation of the prose description. [8].

$\text{Exp}_{\text{ARKG}, \mathcal{A}}^{\text{ks}}(\lambda):$

```

1  pp ← Setup(1λ)
2   $\mathcal{L}_{\text{keys}}, \mathcal{L}_{\text{sk}'} \leftarrow \emptyset$ 
3   $(\text{pk}_0, \text{sk}_0) \leftarrow \text{KGen}(\text{pp})$ 
4   $(\text{pk}^*, \text{sk}^*, \text{rec}^*) \leftarrow \text{\$ } \mathcal{A}^{\mathcal{O}_{\text{pk}'}, \mathcal{O}_{\text{sk}'}}(\text{pp}, \text{pk}_0)$ 
5   $\text{sk}' \leftarrow \text{DeriveSK}(\text{pp}, \text{sk}_0, \text{rec}^*)$ 
6  return  $\llbracket (\text{pk}^*, \text{rec}^*) \in \mathcal{L}_{\text{keys}} \wedge \check{(\text{pk}^*, \text{sk}^*)} = 1 \wedge \check{(\text{pk}^*, \text{sk}')} = 1 \wedge \text{rec}^* \notin \mathcal{L}_{\text{sk}'} \rrbracket$ 

 $\mathcal{O}_{\text{pk}'}(\text{pp}, \text{pk}_0, \cdot)$  on input aux:
7   $(\text{pk}', \text{rec}) \leftarrow \text{\$ } \text{DerivePK}(\text{pp}, \text{pk}_0, \text{aux})$ 
8   $\mathcal{L}_{\text{keys}} \leftarrow \mathcal{L}_{\text{keys}} \cup (\text{pk}', \text{rec})$ 
9  return  $(\text{pk}', \text{rec})$ 

 $\mathcal{O}_{\text{sk}'}(\cdot)$  on input rec:
10  $\text{sk}' \leftarrow \text{DeriveSK}(\text{pp}, \text{sk}_0, \text{rec})$ 
11  $\mathcal{L}_{\text{sk}'} \leftarrow \mathcal{L}_{\text{sk}'} \cup \text{rec}$ 
12 if  $(\cdot, \text{rec}) \notin \mathcal{L}_{\text{keys}}$  : abort
13 return  $\text{sk}'$ 

```

**Fig. 5.** SK-Security as defined in [8] (honest variants).<sup>3</sup>

*Honest vs. Malicious Security.* For this security definition, the term *honest* refers to the first requirement in Line 6 in Fig. 5, which enforces that  $(\text{pk}^*, \text{rec}^*)$  must be in  $\mathcal{L}_{\text{keys}}$ , i.e., that the tuple was honestly generated via *DerivePK*. This same requirement is mirrored in our check that  $(\text{pk}', \text{aux}) \in \mathcal{L}_{\text{keys}}$  in Line 5 in Fig. 2. In the malicious setting, this requirement is dropped, thus giving the adversary more leeway. However, Frymann et al. themselves note in [7], that this may be “too strong for many applications”.

*Weak vs. Strong Security* Frymann et al. have another dimension of security in their definition, which they term *weak* and *strong* security, respectively. The distinction is made along the presence of the highlighted oracle  $\mathcal{O}_{\text{sk}'}$  in Line 4 and the highlighted condition  $\text{rec}^* \notin \mathcal{L}_{\text{sk}'}$  in Line 6 in Fig. 5. If it is present (the strong setting), the adversary is required to output a valid secret key  $\text{sk}^*$  for a pair  $(\text{pk}', \text{rec}^*)$  for which it has not already learned a secret key via a query to  $\mathcal{O}_{\text{sk}'}$ . In the weak setting, the adversary may not learn derived secret keys.

**Delineation.** We argue that the formalization by Frymann et al. is inadequate in the context of ARKG within FIDO2. An adversary that can output the triple of values specified by the security game cannot complete a malicious account recovery. In other words, a successful attacker against the security notions is unable to mount an actual attack in the real world. This problem persists independent of the weak/strong and honest/malicious definitions.

We find that this notion of security does not capture the real-world setting, where an adversary is already successful if it can forge a signature during the recovery process and thus gain access to the user’s account. Our AUTH-security notion, which takes the place of key security, does not require the adversary to output a valid secret key to win, it only requires that the adversary can sign the challenge provided by the relying party during the recovery mechanism. Analogously to the relevant security results by Frymann et al., our definition aligns

with their *honest* setting, since an adversary can only be considered successful in account recovery if it can convince a relying party to successfully verify the signature under the honestly generated public key it has stored as the recovery credential for the user.

With regards to the strong vs. weak setting of Frymann et al. our AUTH-security definition provides capabilities to an adversary roughly comparable to the strong setting. The provided oracles correlate to the case, where there is virtually unlimited access to the backup authenticator with all its capabilities, except of course for trivial attacks. Such an attacker is very powerful and a weaker notion could be defined, but as observed in Remark 3 relaxing the notion would not ease any of the requirements for the underlying primitives.

### A.2 Public-Key Unlinkability

The security notion of unlinkability has the goal of capturing an adversary’s inability to identify multiple derived public keys belonging to the same backup authenticator.

Figure 6 gives a complete pseudocode description of the so-called public-key unlinkability as proposed by Frymann et al. [8]. Essentially, the adversary is given the long-term public key  $pk_{BA}$  of a backup authenticator and may then receive key pairs, which, depending on a hidden bit  $b$ , are either derived from this long-term public key or sampled independently from the key-pair distribution  $D$ .

$\text{Exp}_{\text{ARKG}}^{\text{pku}}(\mathcal{A}):$	$\mathcal{O}_{pk'}^b(b, pk_{BA}, sk_{BA})$ with no input:
<pre> 1  pp ← Setup(<math>1^\lambda</math>) 2  (<math>pk_0, sk_0</math>) ← \$ KGen(pp) 3  <math>b \leftarrow \\$ \{0, 1\}</math> 4  <math>b' \leftarrow \\$ \mathcal{A}_{pk'}^{O_{pk'}^b}(\text{pp}, pk_0)</math> 5  <b>return</b> <math>\llbracket b = b' \rrbracket</math>                 </pre>	<pre> 6  <b>if</b> <math>b = 0</math> 7    (<math>pk', \text{rec}</math>) ← \$ DerivePK(<math>pk_{BA}, aux</math>) 8    <math>sk' \leftarrow</math> DeriveSK(<math>sk_{BA}, \text{rec}</math>) 9  <b>else</b> 10   (<math>pk', sk'</math>) ← \$ D 11  <b>return</b> (<math>pk', sk'</math>)                 </pre>

**Fig. 6.** Public-key unlinkability as defined in [8].<sup>4</sup>

**Delineation.** On its own, the definition by Frymann et al. makes sense to formalize that derived public keys do not leak from which long-term public key they were derived. However, one can show that an ARKG scheme that satisfies public-key unlinkability under Frymann et al.’s definition can output trivially linkable keys, which is also observed in [22] This is because the definition above does not take into account the actual information an adversary has at its disposal.

During registration of derived public keys  $pk'$ , not only is  $pk'$  sent over the wire but also the credential information  $\text{rec}$ , which the relying party also sends

<sup>4</sup> We note that the pseudocode description of  $\mathcal{O}_{pk'}^b$  has not been given before and thus corresponds merely to our interpretation of the prose description. [8]. In particular, it is underspecified how  $aux$  in Line 7 is chosen.

back over an insecure channel when account recovery is triggered. This  $\text{rec}$  may contain  $\text{pk}_{\text{BA}}$ : there is nothing in the construction per se that forbids this. But then public keys derived from this  $\text{pk}_{\text{BA}}$  are all trivially linkable by the adversary.

We want to stress that the linkability is not always as easy to spot (or prevent) as in this example. Especially in the (post-quantum) KEM setting it is not always guaranteed that schemes that provide standard indistinguishability of ciphertexts do not leak information on the public key for which the encapsulation took place. As we show in our results, only KEMs that satisfy ANON-CCA security do provide this guarantee and thus any KEM-based ARKG schemes must ensure this property to provide unlinkability of derived public keys in the presence of recovery information, which we term simply unlinkability.

We thus opted to define unlinkability as a game where the adversary gets to see two long-term public keys  $\text{pk}_{\text{BA}}^0$  and  $\text{pk}_{\text{BA}}^1$  and can as a challenge derive a public key and recovery information with auxiliary information of its choice. Multiple queries would also be easily supported due to a hybrid argument. Furthermore, the adversary may query recovery information of its choice (not the challenge) and let the oracle derive the secret key either from  $\text{sk}_{\text{BA}}^0$  or  $\text{sk}_{\text{BA}}^1$ .

## B Definitions

This appendix will introduce definitions for common building blocks used throughout this work.

### B.1 Key Encapsulation Mechanisms

A KEM scheme is a public key based scheme to generate and communicate a shared secret over an unsecure channel. The primary use case for KEMs is key establishment. KEMs are non-interactive, meaning only one party can contribute randomness. The length of the key as well as the ciphertext are dependent on the security parameter and can be expressed as  $\Gamma(\lambda)$  for the length of the key and  $\Theta(\lambda)$  for the length of the ciphertext. The receiving party cannot influence on the key generation process and has to trust the generating party to use adequate randomness. A key encapsulation scheme KEM consists of three algorithms  $\text{KEM} = (\text{KGen}, \text{Encaps}, \text{Decaps})$ .

$\text{KGen}$  is a probabilistic algorithm that takes the security parameter  $\lambda$  as input and probabilistically outputs a key pair  $(\text{pk}, \text{sk})$ .  $\text{Encaps}$  is a probabilistic algorithm and takes as input a public key  $\text{pk}$ , where  $\text{pk} \leftarrow \text{KGen}(1^\lambda)$ , and outputs a key  $k$  as well as a ciphertext  $c$ . The ciphertext  $c$  encapsulates the key  $k$ .  $\text{Decaps}$  is a deterministic algorithm and takes a secret key  $\text{sk}$  and a ciphertext  $c$  as input, outputting either a key  $k$  or  $\perp$  to indicate failure.

**Definition 1 (Correctness of KEMschemes).** A key encapsulation scheme  $\text{KEM} = (\text{KGen}, \text{Encaps}, \text{Decaps})$  is  $\delta$ -correct, if for all  $(\text{sk}, \text{pk}) \leftarrow_{\$} \text{KGen}(1^\lambda)$  we have  $\Pr[\text{Decaps}(\text{sk}, c) = k : (c, k) \leftarrow_{\$} \text{Encaps}(\text{pk})] \geq 1 - \delta$ . If  $\delta = 0$  holds, the scheme is called perfectly correct.

The security of a KEM scheme is defined over the indistinguishability of derived keys and random keys. A challenger is provided a triple  $(\text{pk}, c, k_b)$ , where  $c$  is output by  $(c, k) \leftarrow \text{Encaps}(\text{pk})$  and  $k_b$  is either sampled uniformly as  $\{0, 1\}^{\Gamma(\lambda)}$  or the actual key, which was output by the encapsulation algorithm. A challenger is successful if it can decide whether the given  $k_b$  is randomly sampled or generated by the encapsulation algorithm with non-negligible probability.

**Definition 2 (IND-ATKsecurity of KEMschemes).** Given the security game in Fig. 7, a key encapsulation scheme  $\text{KEM} = (\text{KGen}, \text{Encaps}, \text{Decaps})$  is IND-ATK secure for  $\text{ATK} \in \{\text{CPA}, \text{CCA}\}$ , if the advantage

$$\text{Adv}_{\text{KEM}, \mathcal{A}}^{\text{ind-atk}}(\lambda) := \Pr[\text{Exp}_{\text{KEM}, \mathcal{A}}^{\text{ind-atk}}(\lambda) = 1]$$

is negligible in the security parameter  $\lambda$  for any QPT adversary  $\mathcal{A}$ .

An additional property some KEM schemes achieve is *anonymity*. Intuitively, anonymity requires that the ciphertext obtained during encapsulation does not leak any information on the public key used during the encapsulation operation.

**Definition 3 (Anonymity of KEM Schemes)** Given the security game in Fig. 8, a key encapsulation scheme  $\text{KEM}$ , is ANON-ATK secure with  $\text{ATK} \in \{\text{CPA}, \text{CCA}\}$ , if the advantage

$$\text{Adv}_{\text{KEM}, \mathcal{A}}^{\text{ANON-ATK}}(\lambda) := |\Pr[\text{Exp}_{\text{KEM}, \mathcal{A}}^{\text{ANON-ATK}}(\lambda) = 1] - \frac{1}{2}|$$

is negligible in the security parameter  $\lambda$  for any QPT adversary  $\mathcal{A}$ .

<u><math>\text{Exp}_{\text{KEM}, \mathcal{A}}^{\text{ind-cpa}}(\lambda)</math>:</u>	<u><math>\text{Exp}_{\text{KEM}, \mathcal{A}}^{\text{ind-cca}}(\lambda)</math>:</u>	<u><math>O_{\text{Dec}}(\cdot)</math> on input <math>c</math>:</u>
1 $(\text{pk}, \text{sk}) \leftarrow_{\$} \text{KGen}(1^\lambda)$	1 $(\text{pk}, \text{sk}) \leftarrow_{\$} \text{KGen}(1^\lambda)$	7 <b>if</b> $c = c^*$
2 $k_0 \leftarrow_{\$} \mathcal{K}$	2 $k_0 \leftarrow_{\$} \mathcal{K}$	8 <b>return</b> $\perp$
3 $(c^*, k_1) \leftarrow_{\$} \text{Encaps}(\text{pk})$	3 $(c^*, k_1) \leftarrow_{\$} \text{Encaps}(\text{pk})$	9 <b>return</b> $\text{Decaps}(\text{sk}, c)$
4 $b \leftarrow_{\$} \{0, 1\}$	4 $b \leftarrow_{\$} \{0, 1\}$	
5 $b' \leftarrow_{\$} \mathcal{A}(\text{pk}, c^*, k_b)$	5 $b' \leftarrow_{\$} \mathcal{A}^{O_{\text{Dec}}(\cdot)}(\text{pk}, c^*, k_b)$	
6 <b>return</b> $\llbracket b' = b \rrbracket$	6 <b>return</b> $\llbracket b' = b \rrbracket$	

**Fig. 7.** Game definition for IND-ATK security of key encapsulation mechanisms with  $\text{ATK} \in \{\text{CPA}, \text{CCA}\}$

$\text{Exp}_{\text{KEM}, \mathcal{A}}^{\text{anon-cpa}}(\lambda)$ :	$\text{Exp}_{\text{KEM}, \mathcal{A}}^{\text{anon-cca}}(\lambda)$ :	$\text{O}_{\text{Dec}}(\cdot, \cdot)$ on input $id, c$ :
1 $b \leftarrow_{\$} \{0, 1\}$	1 $b \leftarrow_{\$} \{0, 1\}$	7 <b>if</b> $c = c^*$
2 $(pk_0, sk_0) \leftarrow_{\$} \text{KGen}(1^\lambda)$	2 $(pk_0, sk_0) \leftarrow_{\$} \text{KGen}(1^\lambda)$	8 <b>return</b> $\perp$
3 $(pk_1, sk_1) \leftarrow_{\$} \text{KGen}(1^\lambda)$	3 $(pk_1, sk_1) \leftarrow_{\$} \text{KGen}(1^\lambda)$	9 $k = \text{Decaps}(sk_{id}, c)$
4 $(c^*, k^*) \leftarrow_{\$} \text{Encaps}(pk_b)$	4 $(c^*, k^*) \leftarrow_{\$} \text{Encaps}(pk_b)$	10 <b>return</b> $k$
5 $b' \leftarrow_{\$} \mathcal{A}(pk_0, pk_1, c^*, k^*)$	5 $b' \leftarrow_{\$} \mathcal{A}^{\text{O}_{\text{Dec}}}(\text{pk}_0, \text{pk}_1, c^*, k^*)$	
6 <b>return</b> $\llbracket b = b' \rrbracket$	6 <b>return</b> $\llbracket b = b' \rrbracket$	

**Fig. 8.** Game definition for ANON-ATK anonymity of key encapsulation mechanisms with  $\text{ATK} \in \{\text{CPA}, \text{CCA}\}$

### B.2 Digital Signatures

A digital signature scheme is a public-key scheme that can be used to generate publicly verifiable signatures. It is defined as a triple of PPT algorithms  $\text{Sig} = (\text{KGen}, \text{Sign}, \text{Vrfy})$ .

$\text{KGen}$  takes as input the security parameter  $\lambda$  and outputs a key pair  $(pk, sk)$ .  $\text{Sign}$  takes as input a secret key  $sk$  and a message  $m$ , and computes a signature  $\sigma$  on the message  $m$ .  $\text{Vrfy}$  is used to verify signatures. It takes as input a public key  $pk$ , a signature  $\sigma$ , and a message  $m$ . The output is 1, if  $\sigma$  is a valid signature for the message  $m$  under the public key  $pk$ , otherwise it returns 0.

**Definition 4** *A digital signature scheme  $\text{Sig} = (\text{KGen}, \text{Sign}, \text{Vrfy})$  is correct, if  $\Pr[0 \leftarrow \text{Vrfy}(pk, \sigma, m) : (pk, sk) \leftarrow_{\$} \text{KGen}(1^\lambda), \sigma \leftarrow_{\$} \text{Sign}(sk, m)]$  is negligible in the security parameter  $\lambda$ .*

Security of signature schemes is defined over the notion of unforgeability. For the basic notion of existential unforgeability under chosen message attack (EUF-CMA) we require an adversary with access to a signing oracle to be unable to forge a signature for a message not previously queried to the oracle. This notion is formalized in the following definition

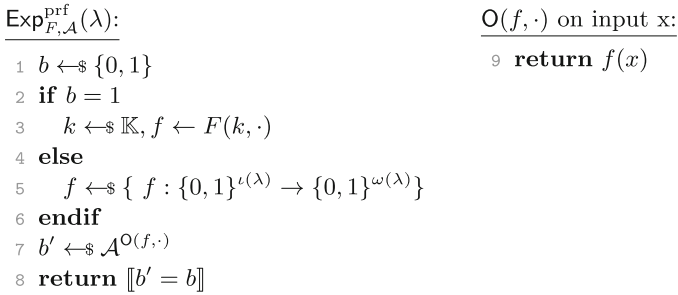
**Definition 5 (EUF-CMA security of digital signature schemes)** *Given the security game in Fig. 9, a digital signature scheme  $\text{Sig} = (\text{KGen}, \text{Sign}, \text{Vrfy})$  is EUF-CMA secure, if  $\text{Adv}_{\text{Sig}, \mathcal{A}}^{\text{euf-cma}}(\lambda) := \Pr[\text{Exp}_{\text{Sig}, \mathcal{A}}^{\text{euf-cma}}(\lambda) = 1]$  is negligible in the security parameter  $\lambda$  for any QPT adversaries  $\mathcal{A}$ .*

$\text{Exp}_{\text{Sig}, \mathcal{A}}^{\text{euf-cma}}(\lambda)$ :	$\text{O}_{\text{Sign}}(sk, \cdot)$ on input $m$ :
1 $(pk, sk) \leftarrow_{\$} \text{KGen}(1^\lambda)$	5 $\sigma \leftarrow_{\$} \text{Sign}(sk, m)$
2 $\mathcal{L}_m \leftarrow \emptyset$	6 $\mathcal{L}_m \leftarrow \mathcal{L}_m \cup \{m\}$
3 $(m', \sigma') \leftarrow_{\$} \mathcal{A}^{\text{O}_{\text{Sign}}(sk, \cdot)}(pk)$	7 <b>return</b> $\sigma$
4 <b>return</b> $\llbracket \text{Vrfy}(pk, \sigma', m') \wedge m' \notin \mathcal{L}_m \rrbracket$	

**Fig. 9.** Game definition for EUF-CMA security of signature schemes

### B.3 PRF Security

**Definition 6 (PRF Security)** Let  $F : \{0, 1\}^{\kappa(\lambda)} \times \{0, 1\}^{\iota(\lambda)} \rightarrow \{0, 1\}^{\omega(\lambda)}$  be an efficient keyed function with key length  $\kappa(\lambda)$ , input length  $\iota(\lambda)$  and output length  $\omega(\lambda)$ . Given the security experiment in Fig. 10, a PRF is secure, if for all QPT adversaries  $\mathcal{A}$  the following holds  $\text{Adv}_{F,\mathcal{A}}^{\text{prf}}(\lambda) := |\Pr [\text{Exp}_{F,\mathcal{A}}^{\text{prf}}(\lambda) = 1] - \frac{1}{2}|$



**Fig. 10.** Game definition for PRF security of a function  $F$

## References

1. Barbosa, M., Boldyreva, A., Chen, S., Warinschi, B.: Provable security analysis of FIDO2. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part III. LNCS, vol. 12827, pp. 125–156. Springer, Cham, Virtual Event (Aug 2021). [https://doi.org/10.1007/978-3-030-84252-9\\_5](https://doi.org/10.1007/978-3-030-84252-9_5)
2. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Berlin, Heidelberg (Dec 2001). [https://doi.org/10.1007/3-540-45682-1\\_33](https://doi.org/10.1007/3-540-45682-1_33)
3. Bindel, N., Cremers, C., Zhao, M.: FIDO2, CTAP 2.1, and WebAuthn 2: Provable Security and Post-Quantum Instantiation. In: IEEE Symposium on Security and Privacy (SP). pp. 674–693 (2023)
4. Bindel, N., Gama, N., Guasch, S., Ronen, E.: To attest or not to attest, this is the question - provable attestation in FIDO2. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part VI. LNCS, vol. 14443, pp. 297–328. Springer, Singapore (Dec 2023). [https://doi.org/10.1007/978-981-99-8736-8\\_10](https://doi.org/10.1007/978-981-99-8736-8_10)
5. Bradley, J., Hodges, J., Jones, M.B., Kumar, A., Lindemann, R., Verrept, J., Antoine, M., Bharadwaj, V., Birgisson, A., Brand, C., Czeskis, A., Duboucher, T., Ehrensward, J., Ploch, M.J., Powers, A., Armstrong, C., Georgantas, K., Kaczmarczyk, F., Satragno, N., Sung, N.: Client to Authenticator Protocol (CTAP) (Jun 2022), <https://fidoalliance.org/specs/fido-v2.1-ps-20210615/fido-client-to-authenticator-protocol-v2.1-ps-errata-20220621.html>
6. Brendel, J., Fischlin, M., Günther, F., Janson, C., Stebila, D.: Towards post-quantum security for Signal’s X3DH handshake. In: Dunkelman, O., Jr., M.J.J., O’Flynn, C. (eds.) SAC 2020. LNCS, vol. 12804, pp. 404–430. Springer, Cham (Oct 2020). [https://doi.org/10.1007/978-3-030-81652-0\\_16](https://doi.org/10.1007/978-3-030-81652-0_16)



7. Frymann, N., Gardham, D., Manulis, M.: Asynchronous remote key generation for post-quantum cryptosystems from lattices. In: 2023 IEEE 8th European Symposium on Security and Privacy (EuroSP). pp. 928–941. IEEE Computer Society, Los Alamitos, CA, USA (jul 2023)
8. Frymann, N., Gardham, D., Kiefer, F., Lundberg, E., Manulis, M., Nilsson, D.: Asynchronous remote key generation: An analysis of yubico’s proposal for W3C WebAuthn. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 939–954. ACM Press (Nov 2023).<https://doi.org/10.1145/3372297.3417292>
9. Frymann, N., Gardham, D., Manulis, M.: Unlinkable delegation of WebAuthn credentials. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) ESORICS 2022, Part III. LNCS, vol. 13556, pp. 125–144. Springer, Cham (Sep 2022).[https://doi.org/10.1007/978-3-031-17143-7\\_7](https://doi.org/10.1007/978-3-031-17143-7_7)
10. Frymann, N., Gardham, D., Manulis, M., Nartz, H.: Generalised asynchronous remote key generation for pairing-based cryptosystems. In: Applied Cryptography and Network Security: 21st International Conference, ACNS 2023, Kyoto, Japan, June 19–22, 2023, Proceedings, Part I. p. 394–421. Springer-Verlag, Berlin, Heidelberg (2023)
11. Guan, J., Li, H., Ye, H., Zhao, Z.: A formal analysis of the FIDO2 protocols. In: Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W. (eds.) ESORICS 2022, Part III. LNCS, vol. 13556, pp. 3–21. Springer, Cham (Sep 2022).[https://doi.org/10.1007/978-3-031-17143-7\\_1](https://doi.org/10.1007/978-3-031-17143-7_1)
12. Hanzlik, L., Loss, J., Wagner, B.: Token meets wallet: Formalizing privacy and revocation for FIDO2. In: 2023 IEEE Symposium on Security and Privacy. pp. 1491–1508. IEEE Computer Society Press (May 2023).<https://doi.org/10.1109/SP46215.2023.10179373>
13. Harell, C.: Yubikeys, passkeys and the future of modern authentication (03 2022), <https://www.yubico.com/blog/passkeys-and-the-future-of-modern-authentication/>
14. Hodges, J., Jones, J., Jones, M.B., Kumar, A., Lundberg, E., Bradley, J., Brand, C., Langley, A., Mandyam, G., Satragno, N., Steele, N., Tan, J., Weedon, S., West, M., Yasskin, J.: Web Authentication: An API for accessing Public Key Credentials - Level 3 (Apr 2021), <https://www.w3.org/TR/webauthn-3>
15. Hülsing, A., Bernstein, D.J., Dobraunig, C., Eichlleder, M., Fluhrer, S., Gazdag, S.L., Kampanakis, P., Kölbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., Aumasson, J.P., Westerman, B., Beullens, W.: SPHINCS<sup>+</sup>. Tech. rep., National Institute of Standards and Technology (2022), available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>
16. Lundberg, E., Nielsson, D.: WebAuthn Recovery Extension (2019), <https://github.com/Yubico/webauthn-recovery-extension>
17. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D., Bai, S.: CRYSTALS-DILITHIUM. Tech. rep., National Institute of Standards and Technology (2022), available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>
18. Maram, V., Xagawa, K.: Post-quantum anonymity of Kyber. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part I. LNCS, vol. 13940, pp. 3–35. Springer, Cham (May 2023).[https://doi.org/10.1007/978-3-031-31368-4\\_1](https://doi.org/10.1007/978-3-031-31368-4_1)
19. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Tech. rep., National Institute of Standards and Technology (2022), available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>

20. Schwabe, P., Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Seiler, G., Stehlé, D., Ding, J.: CRYSTALS-KYBER. Tech. rep., National Institute of Standards and Technology (2022), available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>
21. Shikhar, A.: Charting an Accelerated Path Forward for Passwordless Authentication Adoption (03 2022), <https://fidoalliance.org/charting-an-accelerated-path-forward-for-passwordless-authentication-adoption/>
22. Stebila, D., Wilson, S.: Quantum-safe account recovery for webauthn. Cryptology ePrint Archive, Paper 2024/678 (2022)<https://doi.org/10.1145/3634737.3661138>, <https://eprint.iacr.org/2024/678>, to appear at AsiaCCS '24
23. Xagawa, K.: Anonymity of NIST PQC round 3 KEMs. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 551–581. Springer, Cham (May / Jun 2022).[https://doi.org/10.1007/978-3-031-07082-2\\_20](https://doi.org/10.1007/978-3-031-07082-2_20)

# Author Index

## A

Abe, Masayuki 69

## B

Bacho, Renas 104

Basso, Andrea 339

Branco, Pedro 33

Brendel, Jacqueline 465

## C

Castryck, Wouter 272

Chen, Mingjie 272

Clermont, Sebastian 465

## D

Dartois, Pierrick 304, 339

Duparc, Max 396

Dziembowski, Stefan 174

## E

Ebrahimi, Ehsan 207

## F

Feo, Luca De 339

Fischlin, Marc 465

Fouotsa, Tako Boris 396

Friedman, Offir 141

## I

Invernizzi, Riccardo 272

## J

Jarecki, Stanislaw 174

Jiang, Haodong 433

## K

Kedzior, Pawel 174

Kempner, Octavio Perez 69

Krawczyk, Hugo 174

## L

Lai, Russell W. F. 33

Leroux, Antonin 339

Libert, Benoît 3

Lorenzon, Gioella 272

Loss, Julian 104

## M

Macula, Joseph 371

Maino, Luciano 304, 339

Maitra, Monosij 33

Malavolta, Giulio 33

Marmor, Avichai 141

Mutzari, Dolev 141

## N

Nakagawa, Kohei 243, 272

Nanri, Masaya 69

Ngo, Chan Nam 174

## O

Onuki, Hiroshi 243, 272

## P

Pope, Giacomo 304, 339

## R

Rahimi, Ahmadreza 33

Robert, Damien 304, 339

## S

Scaly, Yehonatan C. 141

Spiizer, Yuval 141

Stange, Katherine E. 371

Stern, Gilad 104

## T

Tibouchi, Mehdi 69

## V

Vercauteren, Frederik 272

**W**

Wagner, Benedikt 104

Wesolowski, Benjamin 339

Woo, Ivy K. Y. 33

**X**

Xu, Jiayu 174

**Y**

Yadav, Anshu 207

Yanai, Avishay 141

**Z**

Zhao, Yunlei 433

Zhou, Biming 433