Kai-Min Chung
Yu Sasaki (Eds.)

LNCS 15485

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 II

2 Part II

INTERNATIONAL ASSOCIATION FOR CRYPTOLOGIC RESEARCH

Springer

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 II

Springer

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 978-981-96-0887-4 ISBN 978-981-96-0888-1 (eBook)
https://doi.org/10.1007/978-981-96-0888-1

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 Carpov, 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 Jhih-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

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

Artifact Chair

Rei Ueno	Kyoto University, Japan

Artifact Evaluation Committee

Julien Béguinot	LTCI, 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 II

Digital Signatures

On Security Proofs of Existing Equivalence Class Signature Schemes

Balthazar Bauer[1](✉), Georg Fuchsbauer[2], and Fabian Regen[2]

[1] UVSQ, Paris, France
balthazar.bauer@ens.fr
[2] TU Wien, Vienna, Austria
georg.fuchsbauer@ens.fr, fabian.regen@tuwien.ac.at

Abstract. Equivalence class signatures (EQS; Asiacrypt '14), sign vectors of elements from a bilinear group. Anyone can transform a signature on a vector to a signature on any multiple of that vector; signatures thus authenticate equivalence classes. A transformed signature/message pair is indistinguishable from a random signature on a random message. EQS have been used to efficiently instantiate (delegatable) anonymous credentials, (round-optimal) blind signatures, ring and group signatures, anonymous tokens and contact-tracing schemes, to name a few.

The original EQS construction (J. Crypto '19) is proven secure in the generic group model, and the first scheme from standard assumptions (PKC '18) satisfies a weaker model insufficient for most applications. Two works (Asiacrypt '19, PKC '22) propose applicable schemes that assume trusted parameters. Their unforgeability is argued via a security proof from standard (or non-interactive) assumptions.

We show that their security proofs are flawed and explain the subtle issue. While the schemes might be provable in the algebraic group model (AGM), we instead show that the original construction, which is more efficient and has found applications in many works, is secure in the AGM under a parametrized non-interactive hardness assumption.

Keywords: Equivalence class signatures · flaw in existing analysis · security proof · algebraic group model

1 Introduction

Structure-preserving Signatures (SPS) [AFG+10] are defined over a *bilinear group*, which consists of three groups $(\mathbb{G}_t, +)$, for $t \in \{1, 2, T\}$, of prime order p and a (non-degenerate) bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. In SPS, messages, as well as public verification keys and signatures, consist of elements from $\mathbb{G}_1$ and $\mathbb{G}_2$.

The concept of *SPS on equivalence classes*, or *equivalence class signatures* (EQS) for short, was introduced by Hanser and Slamanig [HS14] and later securely instantiated [Fuc14, FHS19]. EQS have message space $\mathcal{M} = (\mathbb{G}_t^*)^\ell$, for some $t \in \{1,2\}$, $\ell > 1$, where $\mathbb{G}_t^* := \mathbb{G}_t \setminus \{0_t\}$, on which one defines the following equivalence relation:

$$\boldsymbol{M} \sim \boldsymbol{M}' \;:\Leftrightarrow\; \exists\, \mu \in \mathbb{Z}_p^* : \boldsymbol{M}' = \mu \cdot \boldsymbol{M}. \tag{1}$$

EQS provide an additional functionality ChgRep: given a public key pk, a signature σ on $\boldsymbol{M} \in \mathcal{M}$ under pk, and a value $\mu \in \mathbb{Z}_p^*$, ChgRep returns a signature on the message $\mu \cdot \boldsymbol{M}$, without requiring the secret key. A signature on $\boldsymbol{M}$ thus authenticates the entire equivalence class $[\boldsymbol{M}]_\sim$ of $\boldsymbol{M}$ w.r.t. the relation in (1), and ChgRep lets one change the *representative* of that class.

Accordingly, *unforgeability* is defined w.r.t. classes, that is, for all efficient adversaries, given pk and an oracle for signatures on messages $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots$ of their choice, it is infeasible to compute a signature on any $\boldsymbol{M}^*$ for which $\boldsymbol{M}^* \notin [\boldsymbol{M}_1]_\sim \cup [\boldsymbol{M}_2]_\sim \cup \ldots$ In addition, EQS must be *class-hiding*: it is hard to distinguish random message pairs $(\boldsymbol{M}, \boldsymbol{M}')$ with $\boldsymbol{M} \sim \boldsymbol{M}'$ from random pairs $(\boldsymbol{M}, \boldsymbol{M}') \leftarrow\!\!\$\, \mathcal{M} \times \mathcal{M}$, which is equivalent to the decisional Diffie-Hellman (DDH) problem being hard in $\mathbb{G}_t$.

The last security notion is *signature adaptation* , requiring that for any (possibly maliciously generated) public key pk, any $\boldsymbol{M} \in \mathcal{M}$, any σ that verifies on $\boldsymbol{M}$ under pk, and any $\mu \in \mathbb{Z}_p^*$, running $\mathsf{ChgRep}(pk, \boldsymbol{M}, \sigma, \mu)$ returns a uniform element in the set of all valid signatures on $\mu \cdot \boldsymbol{M}$. This notion, together with class-hiding, implies that a malicious signer that is given some $\boldsymbol{M}$ and generates a signature σ on $\boldsymbol{M}$ cannot distinguish the following: either $\sigma' \leftarrow \mathsf{ChgRep}(pk, \boldsymbol{M}, \sigma, \mu)$ and $\mu \cdot \boldsymbol{M}$ for $\mu \leftarrow\!\!\$\, \mathbb{Z}_p^*$; or a uniformly random signature on a message $\boldsymbol{M}' \leftarrow \mathcal{M}$ under pk.

Applications of EQS. Equivalence class signatures have found numerous applications in concepts related to anonymous authentication. The resulting instantiations are particularly efficient, since both messages and signatures can be *rerandomized*, after which they can be given (and verified) "in the clear", where in other constructions they need to be hidden and shown valid using zero-knowledge proofs.

Anonymous Credentials. The first application of EQS was the construction of *attribute-based credentials* [CL03], which let users obtain credentials for a set of attributes, of which they can later selectively disclose any subset. Such *showings* of attributes should be unlinkable and reveal only the disclosed attributes. The EQS-based credential construction [FHS19] is the first for which the communication complexity of showing a credential is independent of the number of disclosed attributes. Moreover, it achieves strong anonymity guarantees even against malicious credential issuers. Slamanig and others added revocation of users [DHS15] and give a scheme that enables outsourcing of sensitive computation to a restricted device [HS21].

"Signatures with flexible public key" [BHKS18] adapt the concept of adaptation within equivalence classes from messages to public keys, and "mercurial signatures" [CL19, CL21, CLP22] let one adapt signatures to equivalent keys and equivalent messages. The initial motivation of mercurial signatures was the construction of (non-interactively) *delegatable* anonymous credentials [BCC+09,Fuc11], which were later improved [MSBM23]. Multi-authority anonymous credentials have also been constructed from mercurial signatures [MBG+23].

Blind Signatures. Building on earlier work [FV10] that uses randomizable zero-knowledge proofs [FP09], another line of research [FHS15,FHKS16] constructs *blind signatures* from EQS. These allow a user to obtain a signature from a signer, who learns nothing about the message nor the signature. These EQS-based schemes do not assume a common reference string, achieve blindness against malicious signers and are round-optimal and thus concurrently secure.

Group Signatures. Derler and Slamanig [DS16] and Clarisse and Sanders [CS20] use EQS to construct very efficient group signatures schemes. The former also added dynamic adding of members [DS18].

Further applications of EQS include *verifiably encrypted signatures* [HRS15], *access-control encryption* [FGKO17], *sanitizable signatures* [BLL+19], privacy-preserving *incentive systems* [BEK+20], *policy-compliant signatures* [BSW23], *e-voting* [Poi23], and many more.

The FHS Scheme. The first EQS scheme [FHS19], to which we will refer as FHS, has signatures in $\mathbb{G}_1^2 \times \mathbb{G}_2$. This is optimal, since any EQS scheme can be transformed into a structure-preserving signature (SPS) scheme without changing the signature format [FHS15], and SPS signatures must have at least 3 group elements [AGHO11]. Concretely, e.g., when instantiating FHS over the BLS curve [BLS04] BLS12-381 [Bow17,SKSW22], which is conjectured to have 128-bit security, an FHS signature is 192 bytes long.

In addition to yielding optimal instantiations of the aforementioned EQS applications, FHS has seen further applications, such as building highly scalable *mix nets* [HPP20]. Benhamouda, Raykova and Seth [BRS23] use FHS for the currently most efficient instantiation of *anonymous counting tokens*; Hanzlik [Han23] has recently used FHS to construct the first *non-interactive* blind signatures on random messages; and Mir et al. [MSBM23] extended the scheme for their practical delegatable credentials. FHS has been also been proposed for authentication of commercial drones [WTSD23], in the context of e-health [ZYY+23], for whistleblowing reporting systems [SYF+23] and e-voting [Poi24].

Furthermore, FHS underlies the *mercurial signature* construction by Crites and Lysyanskaya [CL19], which have themselves found many applications, some of which are: *Protego* [CDLP22], a credential scheme for permissioned blockchains (like Hyperledger Fabric) and *PACIFIC* [GL23], a privacy-preserving contact tracing scheme. Putman and Martin [PM23] use a modification to construct a delegatable credential scheme that lets users selectively delegate attributes.

The major downside of FHS is that the only proof of its unforgeability to date is directly in the (bilinear) generic group model (GGM) [Nec94,Sho97, Mau05,BBG05], which only captures generic attacks (i.e., ones that work in any group). In security games in the GGM, the adversary does not see any actual group elements but is given (random) labels for them; to compute the group operation, the adversary has access to an oracle which, when given two labels of two elements, returns the label of the sum of these elements.

Constructions from Falsifiable Assumptions. A computational hardness assumption is *falsifiable* [Nao03] if the challenger that runs the security game with an adversary can efficiently decide whether the adversary has broken the assumption. The FHS scheme [FHS19] can be considered based on an (interactive and) *non-falsifiable* assumption: namely its unforgeability, justified via a proof in the generic group model (GGM). Recall that to determine whether the adversary broke unforgeability, one needs to check whether the message $\boldsymbol{M}^*$ returned by the adversary is in the same equivalence class as one of the queried messages (in which case the adversary could efficiently compute a signature on $\boldsymbol{M}^*$ via ChgRep). Now, by the *class-hiding* property, this is hard to decide.

The first EQS scheme from standard assumptions, namely Matrix-Diffie-Hellman assumptions [EHK+13], was proposed by Fuchsbauer and Gay [FG18], but the scheme has some drawbacks: its signatures can only be adapted once and it only satisfies a weaker notion called *existential unforgeability under chosen open message attack* (EUF-CoMA): when the adversary makes a signing query, it must provide the discrete logarithms of the components of the queried message. Note that EUF-CoMA *is* efficiently decidable: For simplicity, consider $\ell = 2$ and for all i, let $(m_{i,1}, m_{i,2}) \in (\mathbb{Z}_p^*)^2$ be the adversary's queries (i.e., the logarithms of the components of the queried message $\boldsymbol{M}_i$). Then the message $\boldsymbol{M}^* = (M_1^*, M_2^*)$ returned by the adversary is not in any of the queried classes if and only if $m_{i,2} \cdot M_1^* \neq m_{i,1} \cdot M_2^*$ for all i.

Khalili, Slamanig and Dakhilalian [KSD19] show that the notion of *signature adaption* achieved by the scheme [FG18] must assume honest keys and signatures, which makes it inadequate for most applications. To construct a scheme appropriate for applications with standard-model security, they first propose more syntax modifications: in addition to a signature, the signing algorithm also creates a *tag*, which is required by ChgRep (but not needed for signature verification). As with the previous scheme [FG18], signatures can only be adapted once (which does not affect the considered applications).

Moreover, they consider a trusted setup, which generates a *common reference string* (CRS) in addition to setting up the bilinear group. *Signature adaptation* is then defined w.r.t. honestly generated parameters. This change weakens the anonymity guarantees in applications such as anonymous credentials, which did not require trust assumptions in the original model [FHS19].

Building on an SPS scheme by Gay et al. [GHKP18], Khalili et al. [KSD19] propose an EQS construction in their new model with signatures in $\mathbb{G}_1^8 \times \mathbb{G}_2^9$. Their construction is (claimed to be) secure under the *SXDH* assumption, which states that DDH is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$. Building on this work, Connolly, Lafourcade and Perez-Kempner [CLP22] propose a more efficient scheme (with signatures in $\mathbb{G}_1^9 \times \mathbb{G}_2^4$), which requires as additional assumption *extKer-MDH* [CH20].

A Flaw in the Security Proof of the CRS-Based Schemes. We report a flaw in the security proofs of the two CRS-based schemes [KSD19, CLP22]. In particular, a game hop in the unforgeability proof changes the distribution of the signatures given to the adversary. The change in the adversary's winning probability is then bounded by the advantage of a reduction in solving a computational problem. However, since EQS-unforgeability is not efficiently decidable, the resulting reduction would not be efficient, and the security bound of the underlying problem can thus not be applied. In fact, the authors do specify an efficient reduction, but its winning probability is not the difference of the adversary's winning probabilities.

In more detail, the hop from Game 0 to Game 1 [KSD19, Theorem 2] modifies the way the purported forgery, i.e., the signature on $\boldsymbol{M}^*$ output by the adversary $\mathcal{A}$ is verified. The authors then argue that from a forgery that verifies in Game 0 but not Game 1 (which is a property that can be checked efficiently), a reduction $\mathcal{B}$ can extract a solution to a computational problem (*KerMDH* [MRV16]). From this, the authors deduce that $\mathbf{Adv}_0 - \mathbf{Adv}_1 \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{KerMDH}}$. This reasoning is *correct*, because (though not stated by the authors) $\mathcal{A}$'s view is equally distributed in both games and thus the probability that $\boldsymbol{M}^*$ does not fall in a class of a queried message (which is not efficiently verifiable) is the same.

In contrast, an analogous argument cannot be made for the hop from Game 2 to Game 3. Here the distribution of the signatures output by the signing oracle changes. (Note that we do not claim that the two games are efficiently distinguishable.) Since the games are different, the probability that $\boldsymbol{M}^*$ falls in a queried class can change in arbitrary ways, but, by class-hiding, this is not efficiently detectable. A change can therefore not be leveraged by an efficient reduction. In fact, the constructed reduction $\mathcal{B}_1$ (to their "core lemma", which relies on the computational hardness of *MDDH* [EHK+17]) only checks an (efficiently testable) property of $\mathcal{A}$'s forgery (but not whether $\mathcal{A}$ was successful). Since whether $\boldsymbol{M}^*$ falls in a queried class determines whether the adversary wins, one can therefore not deduce that $\mathbf{Adv}_2 - \mathbf{Adv}_3 \leq \mathbf{Adv}_{\mathcal{B}_1}^{\mathsf{core}}$, as the authors do. We detail our argument in Sect. 3.

The proof of the second CRS-based scheme [CLP22, eprint, Appendix D] is virtually identical, so the same issue arises. The security of both schemes is thus currently unclear. We believe the schemes cannot be proved from non-interactive assumptions in the standard model. They were derived from a signature scheme [GHKP18] built with proof techniques in mind that crucially rely on the winning condition being efficiently checkable, which is the case for signatures but not for EQS.

Unforgeability of FHS in the Algebraic Group Model. A recent result [BFR24] shows it is unlikely that EQS can be constructed from non-interactive (falsifiable) assumptions in the standard model (that is, without assuming a trusted CRS). Concretely, for any EQS scheme Σ, if there is a reduction that breaks a non-interactive computational assumption after running an adversary that breaks unforgeability of Σ, then there exist efficient meta-reductions that either break the assumption or break class-hiding of Σ. For FHS [FHS19], it was already known that it cannot be proved from non-interactive assumptions via an *algebraic* reduction, since this is the case for all 3-element SPS, and thus EQS, schemes [AGO11].

In light of this result, what we can still hope for is an EQS scheme with a security proof in the *algebraic group model* [FKL18], which is a "weaker" idealized model than the generic group model. In contrast to the latter, in the AGM the adversary has access to the group elements, but the adversary is assumed to be *algebraic* in the following sense: whenever it outputs an element Y of a group $\mathbb{G}_t$, for $t \in \{1, 2\}$, it also provides a *representation* $(\alpha_1, \alpha_2, \dots)$ so that $Y = \alpha_1 Y_1 + \alpha_2 Y_2 + \cdots$, where $Y_1, Y_2, \dots$ are the $\mathbb{G}_t$-elements the adversary has previously received.

Our positive result is a security proof of FHS [FHS19] in the algebraic group model. We focus on FHS due to its optimal efficiency and its many applications discussed above. While the CRS-based schemes [KSD19, CLP22] might be salvageable in the AGM, trying to would be moot, as the signatures of the more efficient scheme [CLP22] are more than 4 times longer than for FHS. Moreover, FHS requires no CRS, an assumption that bars some of the applications of EQS.

We reduce unforgeability of FHS to a parametrized assumption related to the *q-strong Diffie-Hellman* assumption in bilinear groups [BB08]. The latter states that given $G_1, xG_1, x^2G_1, \dots, x^qG_1, G_2, xG_2$, where G_i is a generator of $\mathbb{G}_i$ and x is uniform in $\mathbb{Z}_p$, it is hard to find any $c \in \mathbb{Z}_p$ together with $\frac{1}{x+c}G_1$. Boneh and Boyen [BB08] show that if G_1 and G_2 are random generators, this implies security of their *weakly secure* signature scheme, which corresponds to being given a public key xG_2 and signatures $\frac{1}{x+c_i}G_1$ on messages $c_1, \dots, c_q$ and having to find $\frac{1}{x+c}G_1$ for $c \notin \{c_1, \dots, c_q\}$.

Our assumption combines the above two but is weaker in the sense that it would correspond to security only against key-recovery attacks, where the adversary must find x. In particular, we assume that given x^iG_1 for $i = 1, \dots, 2q$ and xG_2 as well as $\frac{1}{x+c_i}G_t$ for random c_i for $i = 1, \dots, q$ and $t = 1, 2$, it must be hard to find x. Following Boneh and Boyen, we show that, assuming random generators, for $q_1 := 3q$ and $q_2 := q + 1$ our assumption is implied by the (q_1, q_2)-"power"-DL assumption [Lip12], which requires finding x when given x^iG_t for $t = 1, 2$ and $i = 1, \dots, q_t$. (We note that separation results [BFL20] show it is implausible that power-DL can be shown from DL.)

When setting up the group for an FHS instantiation, one can simply sample random generators; in this case, our results imply AGM-security under power-DL, which now underlies the majority of AGM proofs in the literature, in particular for zk-SNARK schemes [FKL18, MBKM19, GWC19, CHM+20, RZ21, CFF+21, LSZ22]. (On the other hand, if generators are fixed, we still get security under our new assumption.)

Discussion. One might wonder about the value of a proof in the AGM when we already have a GGM proof. First, the AGM is closer to reality, as the adversary attacks the actual scheme and not an ideal simulation of it like in the GGM; the AGM just restricts how the adversary manipulates group elements, which is enforced in the GGM as well. Given the EQS impossibility result [BFR24], an

AGM proof from a non-interactive assumption is arguably the best one can hope for. (The situation is similar for zk-SNARKs, for which there are impossibility results [GW11], and the AGM has become a common model for security analysis; see citations above.) While our proof may be more complex than the GGM proof [FHS19], we improve the result, since a proof in the AGM from an assumption that holds in the GGM implies security in the GGM. (Conversely, there are hardness assumptions, such as *one-more DL* [BNPS03,BFP21], that hold in the GGM but cannot be shown in the AGM from power-DL [BFL20].)

We thus establish a new state of the art for EQS: There are currently no EQS schemes assuming a trusted CRS with a security proof in the standard model. Moreover, our negative result indicates that new proof techniques would be required, instead of starting from existing standard-model SPS schemes like [GHKP18]. Many applications (blind signatures, credentials, etc.) without semi-honesty assumptions require fully secure EQS (without a CRS), for which FHS is the most efficient scheme and has seen many applications. We improve the security guarantees of FHS.

2 Preliminaries

Notation. Assigning a value x to a variable var is denoted by $\mathit{var} := x$. All algorithms are randomized unless otherwise indicated. By $y \leftarrow \mathcal{A}(x_1, \ldots, x_n)$ we denote the operation of running algorithm $\mathcal{A}$ on inputs $x_1, \ldots, x_n$ and letting y denote the output; by $[\mathcal{A}(x_1, \ldots, x_n)]$ we denote the set of values that have positive probability of being output. If S is a finite set then $x \leftarrow\!\!\$\, S$ denotes picking an element uniformly from S and assigning it to x. For $n \in \mathbb{N}$, we let $[n]$ denote the set $\{1, \ldots, n\}$. For $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{Z}_p^n$, we denote $\mathbf{x} \odot \mathbf{y} = (x_1 y_1, \ldots, x_n y_n)$ the Hadamard product of $\mathbf{x}$ and $\mathbf{y}$.

Polynomials. In our proof the FHS scheme in Sect. 4 we will make extensive use of multivariate polynomials in $\mathbb{Z}_p[\mathrm{X}_1, \ldots, \mathrm{X}_n]$, for prime p, and use the following two lemmas.

Lemma 1 (Schwartz-Zippel). *Let p be prime and let $\mathsf{P} \in \mathbb{Z}_p[\mathrm{X}_1, \ldots, \mathrm{X}_n]$ be a non-zero polynomial of total degree d. Then*

$$\Pr_{r_1, \ldots, r_n \leftarrow\!\!\$\, \mathbb{Z}_p^*}[\mathsf{P}(r_1, \ldots, r_n) = 0] \leq \frac{d}{p-1} \ .$$

The next lemma [BFL20, Lemma 2.1] has become a standard tool in AGM proofs. It implies that when embedding an indeterminate Y (which will represent the solution of a computational problem) into many indeterminates $\mathrm{X}_1, \ldots, \mathrm{X}_n$ of an adversarially chosen non-zero polynomial P by "randomizing" Y as $\mathrm{X}_i := z_i \mathrm{Y} + v_i$ for random z_i, v_i then the polynomial $\mathsf{P}'(\mathrm{Y}) := \mathsf{P}(z_1 \mathrm{Y} + v_1, \ldots, z_n \mathrm{Y} + v_n)$ will be non-zero with overwhelming probability. (This relies on the fact that the values z_i are perfectly hidden from the adversary's view.)

Lemma 2. *Let* P *be a non-zero multivariate polynomial in* $\mathbb{Z}_p[\mathrm{X}_1, \ldots, \mathrm{X}_n]$ *of total degree d. If we define* $\mathsf{Q}(\mathrm{Y}) \in \big(\mathbb{Z}_p[\mathrm{Z}_1, \ldots, \mathrm{Z}_m, \mathrm{V}_1, \ldots, \mathrm{V}_n]\big)[\mathrm{Y}]$ *as*

$$\mathsf{Q}(\mathrm{Y}) := \mathsf{P}\big(\mathrm{Z}_1\mathrm{Y} + \mathrm{V}_1, \ldots, \mathrm{Z}_n\mathrm{Y} + \mathrm{V}_n\big)\ ,$$

then the coefficient of maximal degree of Q *is a polynomial in* $\mathbb{Z}_p[\mathrm{Z}_1, \ldots, \mathrm{Z}_n]$ *of degree d.*

Bilinear Groups. EQS schemes are defined over an (asymmetric) bilinear group $grp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G_1, G_2, e)$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are (additively denoted) groups of prime order p, G_1 and G_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, resp., and $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a bilinear map so that $G_T := e(G_1, G_2)$ generates $\mathbb{G}_T$. For $t \in \{1, 2, T\}$, we let $\mathbb{G}_t^* := \mathbb{G}_t \setminus \{0_t\}$. We assume that there exists a probabilistic polynomial-time (p.p.t.) algorithm BGGen, which on input 1^λ, the security parameter in unary, returns the description of a bilinear group grp so that the bit length of p is λ.

Following the examined work [KSD19], we use "implicit" representation of group elements: for $\mathbf{A} = (a_{i,j})_{i,j} \in \mathbb{Z}_p^{m\times n}$ and $t \in \{1, 2, T\}$, we let $[\mathbf{A}]_t$ denote the matrix $(a_{i,j}G_t)_{i,j} \in \mathbb{G}_t^{m\times n}$ and define $e([\mathbf{A}]_1, [\mathbf{B}]_2)$ as $[\mathbf{AB}]_T$, which can be computed efficiently. We use upper-case slanted font $G, \boldsymbol{G}$ to denote group elements and vectors of group elements and use $a, \mathbf{a}, \mathbf{A}$ to denote scalars, vectors and matrices of elements from $\mathbb{Z}_p$.

EQS. An *equivalence class signature (EQS) scheme* Σ specifies an algorithm $\mathsf{ParGen}(1^\lambda)$, which on input the security parameter returns general parameters par, which specify a bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G_1, G_2, e)$. $\mathsf{KeyGen}(par, 1^\ell)$, on input the parameters and the message length $\ell > 1$, returns a key pair (sk, pk), which defines the message space $\mathcal{M} := (\mathbb{G}_t^*)^\ell$ for a fixed $t \in \{1, 2\}$. The message space is partitioned into equivalence classes by the following relation for $\boldsymbol{M}, \boldsymbol{M}' \in \mathcal{M}$:

$$\boldsymbol{M} \sim \boldsymbol{M}' \;:\Leftrightarrow\; \exists\, \mu \in \mathbb{Z}_p^* : \boldsymbol{M}' = \mu \cdot \boldsymbol{M}. \tag{1}$$

A *tag-based* EQS scheme [KSD19] moreover consists of the following algorithms:

- $\mathsf{Sign}(sk, \boldsymbol{M})$, on input a secret key and a message $\boldsymbol{M} \in \mathcal{M}$, returns a signature σ and (possibly) a tag τ.
- $\mathsf{ChgRep}(pk, \boldsymbol{M}, (\sigma, \tau), \mu)$, on input a public key, a message $\boldsymbol{M} \in \mathcal{M}$, a signature σ (and possibly a tag τ) on $\boldsymbol{M}$, as well as a scalar $\mu \in \mathbb{Z}_p^*$, returns a signature σ' on the message $\mu \cdot \boldsymbol{M}$.
- $\mathsf{Verify}(pk, \boldsymbol{M}, (\sigma, \tau))$ is deterministic and, on input a public key, a message $\boldsymbol{M} \in \mathcal{M}$, a signature σ (and possibly a tag τ), returns a bit indicated acceptance.

Sign and ChgRep must generate valid signatures, as defined next.

Definition 1. *An EQS scheme is* ***correct*** *if for all* $\lambda \in \mathbb{N}$, $\ell > 1$, *any* $par \in [\mathsf{ParGen}(1^\lambda)]$, $(sk, pk) \in [\mathsf{KeyGen}(par, 1^\ell)]$, $\boldsymbol{M} \in \mathcal{M}$ *and* $\mu \in \mathbb{Z}_p^*$*:*

$$\Pr\left[\mathsf{Verify}\big(pk, \boldsymbol{M}, \mathsf{Sign}(sk, \boldsymbol{M})\big) = 1\right] = 1 \qquad \textit{and}$$
$$\Pr\left[\mathsf{Verify}\big(pk, \mu \cdot \boldsymbol{M}, \mathsf{ChgRep}(pk, \boldsymbol{M}, \mathsf{Sign}(sk, \boldsymbol{M}), \mu)\big) = 1\right] = 1.$$

Unforgeability requires that after receiving the public key and signatures (and tags) on messages of its choice, the adversary cannot produce a valid signature on a message that is not contained in any of the classes of the queried signatures.

Definition 2. *An EQS scheme* Σ *with message length* $\ell > 1$ *is* ***existentially unforgeable under chosen-message attack*** *if*

$$\mathsf{Adv}^{\mathrm{UNF}}_{\Sigma,\mathcal{A}}(\lambda) := \Pr[\mathrm{UNF}_{\Sigma,\mathcal{A}}(\lambda) = 1]$$

is negligible for all p.p.t. adversaries $\mathcal{A}$*, where game* UNF *is defined as follows:*

```
$\mathrm{UNF}_{\Sigma,\mathcal{A}}(\lambda)$
1 : $par \leftarrow \mathsf{ParGen}(1^\lambda)$
2 : $(sk, pk) \leftarrow \mathsf{KeyGen}(par, 1^\ell)$
3 : $Q := \emptyset$
4 : $(\boldsymbol{M}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(pk)$
5 : return $\big(\boldsymbol{M}^* \notin Q \wedge \mathsf{Verify}(pk, \boldsymbol{M}^*, \sigma^*)\big)$
```

```
$\mathcal{O}(\boldsymbol{M})$
1 : $Q := Q \cup [\boldsymbol{M}]_\sim$
2 : return $\mathsf{Sign}(sk, \boldsymbol{M})$
```

where $[\boldsymbol{M}]_\sim := \{\boldsymbol{M}' \in \mathcal{M} \mid \boldsymbol{M} \sim \boldsymbol{M}'\}$ *is the equivalence class of* $\boldsymbol{M}$ *for* $\sim$ *defined in* (1)*.*

A further security requirement is that signatures generated by ChgRep should either be indistinguishable from signatures output by Sign or uniformly random in the space of all valid signatures. As these notions are not relevant for our results, we refrain from stating them and refer to the original work [FHS19].

3 A Flaw in the Security Proofs of KSD19 and CLP22

The proof of unforgeability [KSD19] defines Game 0 as the game UNF from Definition 2 instantiated with their construction as Σ, and, in a series of "hops", the games are gradually modified until Game 6 can only be won with probability $1/p$, even by an unbounded adversary. The difference between the adversary's advantage $\mathbf{Adv}_i$ in winning Game i and its advantage $\mathbf{Adv}_{i+1}$ in winning Game $(i + 1)$ is then bounded. Of these bounds, two depend on the hardness of a computational problem.

Define event N_i as $\boldsymbol{M}^* \notin Q$ when running Game i (where $\boldsymbol{M}^*$ is from $\mathcal{A}$'s output and Q is the union of all classes of queried messages). Moreover, let V_i be the event that when running Game i, we have $\mathsf{Verify}_i(pk, \boldsymbol{M}^*, \sigma^*)$, where Verify_i is how verification of $\mathcal{A}$'s signature is defined in Game i. (The details of Verify_i are not relevant here.) We thus have $\mathbf{Adv}_i = \Pr[\mathrm{N}_i \wedge \mathrm{V}_i]$.

The First Hop. In Game 0 and Game 1 the adversary's view remains the same, and we therefore have $\mathrm{N}_0 = \mathrm{N}_1$. The only thing that changes is that when verifying $\mathcal{A}$'s forgery, which contains group-element vectors $[\mathbf{u}_1^*]_1$ and $[\mathbf{t}^*]_1$, against $pk = ([\mathbf{A}]_2, [\mathbf{K}_0\mathbf{A}]_2, [\mathbf{K}\mathbf{A}]_2)$, instead of checking

$$e([\mathbf{u}_1^*]_1^\top, [\mathbf{A}]_2) - e([\mathbf{t}^*]_1^\top, [\mathbf{K}_0\mathbf{A}]_2) - e([\mathbf{m}^*]_1^\top, [\mathbf{K}\mathbf{A}]_2) = 0,$$

one checks if $\boldsymbol{S} := [\mathbf{u}_1^*]_1 - \mathbf{K}_0^\top[\mathbf{t}^*]_1 - \mathbf{K}^\top[\mathbf{m}^*]_1 = 0$, which implies the above.

We thus have $\mathrm{V}_1 \subseteq \mathrm{V}_0$ and if V_0 occurs but V_1 does not, then $\mathcal{A}$ has found a non-zero vector $\boldsymbol{S}$ in the kernel of $\mathbf{A}$. The authors construct a reduction $\mathcal{B}$ which uses this to break KerMDH [MRV16] in $\mathbb{G}_2$. We have

$$\begin{aligned}
\mathbf{Adv}_0 - \mathbf{Adv}_1 &= \Pr[\mathrm{N}_0 \wedge \mathrm{V}_0] - \Pr[\mathrm{N}_1 \wedge \mathrm{V}_1] \\
&= \Pr[\mathrm{N}_0 \wedge \mathrm{V}_0 \wedge \mathrm{V}_1] + \Pr[\mathrm{N}_0 \wedge \mathrm{V}_0 \wedge \neg\mathrm{V}_1] \\
&\qquad - \Pr[\mathrm{N}_1 \wedge \mathrm{V}_1 \wedge \mathrm{V}_0] - \Pr[\mathrm{N}_1 \wedge \mathrm{V}_1 \wedge \neg\mathrm{V}_0] \\
&= \Pr[\mathrm{N}_0 \wedge \mathrm{V}_0 \wedge \neg\mathrm{V}_1] \quad (\text{since } \mathrm{N}_0 = \mathrm{N}_1 \text{ and } \mathrm{V}_1 \subseteq \mathrm{V}_0) \\
&\leq \Pr[\mathrm{V}_0 \wedge \neg\mathrm{V}_1] \leq \mathbf{Adv}_{\mathcal{B}}^{\mathsf{KerMDH}}.
\end{aligned}$$

Note that for this argument it was essential that N_0, N_1, V_0 and V_1 are all events in the same probability space (which will not be the case in the hop from Game 2 to Game 3).

The Bad Hop. In the hop from Game 2 to Game 3, the distribution of the game changes and thus we do not have $\mathrm{N}_2 = \mathrm{N}_3$ (which is also syntactically meaningless). The authors construct a reduction $\mathcal{B}_1$ which bounds $\Pr[\mathrm{V}_2] - \Pr[\mathrm{V}_3] \leq \mathbf{Adv}_{\mathcal{B}_1}^{\mathrm{core}}$, where the latter is $\mathcal{B}_1$'s probability in winning the game from their "core lemma" [KSD19, Sect. 4.1], which is bounded by breaking another computational problem (Matrix-DDH [EHK+17]). However, it is not clear how to use this to bound the change in advantage from Game 2 to Game 3. We have

$$\begin{aligned}
\mathbf{Adv}_2 - \mathbf{Adv}_3 &= \Pr[\mathrm{N}_2 \wedge \mathrm{V}_2] - \Pr[\mathrm{N}_3 \wedge \mathrm{V}_3] \\
&= \Pr[\mathrm{N}_2 \,|\, \mathrm{V}_2] \cdot \big(\underbrace{\Pr[\mathrm{V}_2] - \Pr[\mathrm{V}_3]}_{(1)}\big) + \big(\underbrace{\Pr[\mathrm{N}_2 \,|\, \mathrm{V}_2] - \Pr[\mathrm{N}_3 \,|\, \mathrm{V}_3]}_{(2)}\big) \cdot \Pr[\mathrm{V}_3].
\end{aligned}$$

So while we can bound (1) by $\mathcal{B}_1$'s advantage of breaking the "core lemma", it is unclear how to bound (2). In particular, N_i is an event that cannot be efficiently checked, and moreover, in contrast to N_0 and N_1, the events N_2 and N_3 are not equivalent, since the adversary's view is different on Game 2 and Game 3.

To show this, we spell out Game i for $i \in \{2, 3\}$ in Fig. 1, where Verify_i denotes how verification is defined in Game i (both Verify_2 and Verify_3 are efficient, but their details not relevant here). Moreover, $\mathcal{D}_1$ is a distribution of matrices from $\mathbb{Z}_p^{2\times 1}$ for which the MDDH assumption must hold; PGen and PPro belong to

```
Game (2 + β)
  $grp \leftarrow \mathsf{BGGen}(1^\lambda)$ ; $ctr := 0$
  $\mathbf{A}_0 \leftarrow\$ \mathcal{D}_1$ ; $\mathbf{A}_1 \leftarrow\$ \mathcal{D}_1$
  $crs \leftarrow \mathsf{PGen}(grp, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1)$
  $par := (grp, [\mathbf{A}_0]_1, [\mathbf{A}_1]_1, crs)$
  $\mathbf{A} \leftarrow\$ \mathcal{D}_1$
  $\mathbf{K}_0 \leftarrow\$ \mathbb{Z}_p^{2\times 2}$ ; $\mathbf{K} \leftarrow\$ \mathbb{Z}_p^{\ell\times 2}$
  $\mathbf{a}^\perp \leftarrow\$ \{\mathbf{a}^\perp \in \mathbb{Z}_p^2 \mid (\mathbf{a}^\perp)^\top \mathbf{A} = 0\}$
  $\mathbf{k}_0 \leftarrow\$ \mathbb{Z}_p^2$ ; $\mathbf{k}_1 \leftarrow\$ \mathbb{Z}_p^2$
  $\mathbf{K}_0 := \mathbf{K}_0 + \mathbf{k}_0(\mathbf{a}^\perp)^\top$
  $pk := ([\mathbf{A}]_2, [\mathbf{K}_0\mathbf{A}]_2, [\mathbf{KA}]_2)$
  $Q := \emptyset$
  $([\mathbf{m}^*]_1, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}(par, pk)$
  return $\big([\mathbf{m}^*]_1 \notin Q$
          $\wedge\ \mathsf{Verify}_i(pk, [\mathbf{m}^*]_1, \sigma^*)\big)$
```

```
$\mathcal{O}([\mathbf{m}]_1)$
  $Q := Q \cup [[\mathbf{m}]_1]_\sim$
  $r_1, r_2 \leftarrow\$ \mathbb{Z}_p$
  $[\mathbf{t}]_1 := [\mathbf{A}_0]_1 r_1$ ; $[\mathbf{w}]_1 := [\mathbf{A}_0]_1 r_2$
  $\Omega \leftarrow \mathsf{PPro}(crs, [\mathbf{t}]_1, r_1, [\mathbf{w}]_1, r_2)$
  $(\Omega_1, \Omega_2, [\mathbf{z}_0]_2, [\mathbf{z}_1]_2, \pi) := \Omega$
  $ctr := ctr + 1$
  $[\mathbf{u}_1]_1 := \mathbf{K}_0^\top [\mathbf{t}]_1 + \mathbf{K}^\top [\mathbf{m}]_1$
          $+ \mathbf{a}^\perp \big(\mathbf{k}_0 + \beta \cdot \mathbf{F}(ctr)\big)^\top [\mathbf{t}]_1$
  $[\mathbf{u}_2]_1 := \mathbf{K}_0^\top [\mathbf{w}]_1$
          $+ \mathbf{a}^\perp \big(\mathbf{k}_0 + \beta \cdot \mathbf{k}_1\big)^\top [\mathbf{w}]_1$
  $\sigma := ([\mathbf{u}_1]_1, \Omega_1, [\mathbf{z}_0]_2, [\mathbf{z}_1]_2, \pi, [\mathbf{t}]_1)$
  $\tau := ([\mathbf{u}_2]_1, \Omega_2, [\mathbf{w}]_1)$
  return $(\sigma, \tau)$
```

Fig. 1. Games 2 and 3 in the unforgeability proof of [KSD19]. Changes w.r.t. game UNF are denoted in gray, the differences between Games 2 and 3 are highlighted in blue. The line in red is our interpretation, since the distribution of $\mathbf{a}^\perp$ is not specified.

a proof system for statements $([\mathbf{t}]_1, [\mathbf{w}]_1)$ which are true if $[\mathbf{t}]_1 = [\mathbf{A}_b]_1 r_1$ and $[\mathbf{w}]_1 = [\mathbf{A}_b]_1 r_2$ for some $b \in \{0,1\}$ and $r_1, r_2 \in \mathbb{Z}_p$ (again, the details are not relevant here); and $\mathbf{F}\colon \mathbb{Z}_p \to \mathbb{Z}_p^2$ is a random function.

To argue that $\mathcal{A}$'s view changes from Game 2 to Game 3, an easy way is to have $\mathcal{A}$ query the signing oracle $\mathcal{O}$ twice on the same (arbitrary) message. For the i-th query, let $r_1^{(i)}$ and $r_2^{(i)}$ be the randomness sampled by $\mathcal{O}$ and let $\mathbf{u}_1^{(i)}, \mathbf{t}^{(i)}, \mathbf{u}_2^{(i)}, \mathbf{w}^{(i)} \in \mathbb{Z}_p^2$ be the logarithms of the respective components returned by $\mathcal{O}$.

Since $\mathbf{A}_0 \in \mathbb{Z}_p^{2\times 1}$ is from a "matrix distribution" [KSD19, Definition 1], it has full rank and is thus non-zero. The value $\mathbf{t}^{(i)} = \mathbf{A}_0 r_1^{(i)}$ thus uniquely determines $r_1^{(i)}$ and $\mathbf{w}^{(i)} = \mathbf{A}_0 r_2^{(i)}$ uniquely determines $r_2^{(i)}$. Let $r_1' := r_1^{(1)} - r_1^{(2)}$ and $r_2' := r_2^{(1)} - r_2^{(2)}$, and thus $\mathbf{t}^{(1)} - \mathbf{t}^{(2)} = \mathbf{A}_0 r_1'$ and $\mathbf{w}^{(1)} - \mathbf{w}^{(2)} = \mathbf{A}_0 r_2'$, and consider these further differences:

$$\mathbf{u}_1' := \mathbf{u}_1^{(1)} - \mathbf{u}_1^{(2)} = \mathbf{K}_0^\top \mathbf{A}_0 r_1' + \mathbf{a}^\perp \mathbf{k}_0^\top \mathbf{A}_0 r_1' + \beta \cdot \mathbf{a}^\perp \big(\mathbf{F}(1)^\top \mathbf{A}_0 r_1^{(1)} - \mathbf{F}(2)^\top \mathbf{A}_0 r_1^{(2)}\big)$$
$$\mathbf{u}_2' := \mathbf{u}_2^{(1)} - \mathbf{u}_2^{(2)} = \mathbf{K}_0^\top \mathbf{A}_0 r_2' + \mathbf{a}^\perp \mathbf{k}_0^\top \mathbf{A}_0 r_2' + \beta \cdot \mathbf{a}^\perp \mathbf{k}_1^\top \mathbf{A}_0 r_2'$$

In Game 2, where $\beta = 0$, we thus have

$$\mathbf{u}_1' r_2' = \mathbf{u}_2' r_1'. \tag{2}$$

On the other hand, for (2) to hold in Game 3, we would have to have

$$\mathbf{a}^{\perp}\big(\mathbf{F}(1)^{\top}\mathbf{A}_0 r_1^{(1)} - \mathbf{F}(2)^{\top}\mathbf{A}_0 r_1^{(2)}\big) r_2' = \mathbf{a}^{\perp}\mathbf{k}_1^{\top}\mathbf{A}_0 r_2' (r_1^{(1)} - r_1^{(2)}),$$

or equivalently

$$\mathbf{a}^{\perp}\big(\underbrace{\mathbf{F}(1)^{\top} r_1^{(1)} - \mathbf{F}(2)^{\top} r_1^{(2)} - \mathbf{k}_1^{\top}(r_1^{(1)} - r_1^{(2)})}_{=:\mathbf{U}^{\top}}\big)\mathbf{A}_0 r_2' = \mathbf{0}. \tag{3}$$

Since $\mathbf{F}(1)$ is independent and uniformly distributed in $\mathbb{Z}_p^2$, the term $\mathbf{U}$ is uniform in $\mathbb{Z}_p^2$, except with negligible probability (when $r_1^{(1)} = 0$). As argued above, $\mathbf{A}_0$ is non-zero and thus $\mathbf{U}^{\top}\mathbf{A}_0$ is uniform in $\mathbb{Z}_p$ (except with negligible probability). The authors [GHKP18, KSD19] do not specify how $\mathbf{a}^{\perp}$ is distributed, but for their last argument in the proof to work, namely that Game 6 can only be won with probability $1/p$ (or with negligible probability), we must have $\mathbf{a}^{\perp} \neq \mathbf{0}$ (with overwhelming probability). Thus for (3) (and thus (2)) to hold, we must either have $\mathbf{a}^{\perp} = \mathbf{0}$ or $\mathbf{U}^{\top}\mathbf{A}_0 = 0$ or $r_2' = 0$, which happens with negligible probability only.

Thus, the view of the adversary changes between Games 2 and 3, and therefore so can its probability of returning a messages that is in the class of a queried message, i.e., we can have that $\Pr[\mathrm{N}_2]$ and $\Pr[\mathrm{N}_3]$ differ by a non-negligible amount. The argument which worked for bounding $\mathbf{Adv}_0 - \mathbf{Adv}_1$ (a reduction that only considers the events V_0 and V_1), and which the authors also apply to bound $\mathbf{Adv}_2 - \mathbf{Adv}_3$, can thus not be made again.

4 The Security of FHS in the AGM

With the FHS EQS scheme remaining the only scheme with some security proof [FHS19], we will strengthen its security guarantees by giving a proof in the *algebraic group model* (AGM) under a parametrized hardness assumption. We start with defining the scheme.

Definition 3 ([FHS19]). *Let* $grp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, \hat{G}, e)$ *be a bilinear group output by* $\mathsf{BGGen} = \mathsf{ParGen}$. *Let* $\ell > 1$ *and* $\mathcal{M} := (\mathbb{G}_1^*)^{\ell}$. *The EQS scheme* ***FHS*** *is defined as follows:*

- $\mathsf{KeyGen}(grp, 1^{\ell})$*: sample* $\boldsymbol{x} \leftarrow_{\$} (\mathbb{Z}_p^*)^{\ell}$, *set* $sk := \boldsymbol{x}$, $pk := \hat{\boldsymbol{X}} = \big(x_1\hat{G}, \ldots, x_{\ell}\hat{G}\big)$.
- $\mathsf{Sign}(\boldsymbol{x}, \boldsymbol{M})$*: sample* $r \leftarrow_{\$} \mathbb{Z}_p^*$ *and return* $\sigma := \Big(r\sum_{i=1}^{\ell} x_i M_i, \frac{1}{r}G, \frac{1}{r}\hat{G}\Big)$.
- $\mathsf{ChgRep}(\hat{\boldsymbol{X}}, \boldsymbol{M}, (Z, R, \hat{R}), \mu)$ *sample* $r \leftarrow_{\$} \mathbb{Z}_p^*$ *and return* $\sigma' := \big(\mu r Z, \frac{1}{r}R, \frac{1}{r}\hat{R}\big)$
- $\mathsf{Verify}(\hat{\boldsymbol{X}}, \boldsymbol{M}, (Z, R, \hat{R}))$*: return* 1 *if and only if*

$$\sum\nolimits_{i=1}^{\ell} e\big(M_i, \hat{X}_i\big) = e\big(Z, \hat{R}\big) \qquad \textit{and} \tag{4}$$

$$e\big(R, \hat{G}\big) = e\big(G, \hat{R}\big)\ . \tag{5}$$

Correctness is immediate (cf. [FHS19]). While, so far, the scheme has only been proven secure in the generic group model, we will give a proof in the AGM.

Definition 4 (EQS-unforgeability in the AGM). *The algebraic unforgeability game* $\mathrm{UNF}^{\mathrm{AGM}}$ *is obtained from the UNF game from Definition 2 with the following changes: whenever the adversary $\mathcal{A}$ outputs an element Y of a group $\mathbb{G}_t$, for $t \in \{1, 2\}$, it also provides a* representation $\boldsymbol{\alpha}$ *s.t.* $Y = \sum \alpha_i Y_i$, *where $\{Y_i\}$ is the set of previously received elements from $\mathbb{G}_t$.*

The (q_1, q_2)-DL assumption [Lip12] in a bilinear group $grp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, \hat{G}, e)$ states that for a randomly sampled $y \leftarrow\!\!\$\, \mathbb{Z}_p$, no efficient adversary, that is given grp as well as $y^i G$ for $i \in [q_1]$ and $y^i \hat{G}$ for $i \in [q_2]$, can find y.

We introduce a variant of this assumption, where in addition to powers of the challenge value y^i (in the form $y^i G_t$), the adversary receives denominators $1/(y + c_i)$ for random known values c_i. This is reminiscent of the assumption corresponding to the *weakly secure* Boneh-Boyen signatures [BB04], which is implied by their *strong Diffie-Hellman* assumption. Analogously, we show that, under similar conditions, our assumption is implied by the standard (q_1, q_2)-DL assumption for appropriate q_1 and q_2.

Definition 5. *The q-**PowDenDL assumption** holds with respect to* BGGen *if* $\mathsf{Adv}^{q\text{-PowDenDL}}_{\mathsf{BGGen},\mathcal{A}}(\lambda) := \Pr[q\text{-PowDenDL}_{\mathsf{BGGen},\mathcal{A}}(\lambda) = 1]$ *is negligible for all p.p.t. adversaries $\mathcal{A}$, where game* q-PowDenDL *is defined as follows:*

```
$q\text{-PowDenDL}_{\mathsf{BGGen},\mathcal{A}}(\lambda)$
1: $grp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, \hat{G}, e) \leftarrow \mathsf{BGGen}(1^\lambda)$
2: $y \leftarrow\!\!\$\, \mathbb{Z}_p$ ; $(c_1, \ldots, c_q) \leftarrow\!\!\$\, \mathbb{Z}_p^q$
3: if $(-y \bmod p) \in \{c_1, \ldots, c_q\}$ then return 1
4: $y' \leftarrow \mathcal{A}\Big(grp, (y^i G)_{i=1}^{2q}, y\hat{G}, \big(\tfrac{1}{y+c_i} G, \tfrac{1}{y+c_i} \hat{G}, c_i\big)_{i=1}^{q}\Big)$
5: return $y = y'$
```

We show that, assuming that BGGen returns random generators, q-PowDenDL is implied by (q_1, q_2)-DL for $q_1 := 3q$ and $q_2 := q + 1$; we follow Boneh and Boyen's proof [BB04] (who for their scheme also assume that generators are randomly sampled).

Lemma 3. *Let q be arbitrary and* BGGen *be such that G and $\hat{G}$ are uniformly random. If $(3q, q+1)$-DL holds then q-PowDenDL holds; concretely, for every $\mathcal{A}$ there exists $\mathcal{B}$ with essentially the same running time such that*

$$\mathsf{Adv}^{(3q,q+1)\text{-DL}}_{\mathsf{BGGen},\mathcal{B}}(\lambda) \geq \mathsf{Adv}^{q\text{-PowDenDL}}_{\mathsf{BGGen},\mathcal{A}}(\lambda) .$$

Proof. Let $\mathcal{A}$ be an adversary against q-PowDenDL. We construct an adversary $\mathcal{B}$ against $(3q, q+1)$-DL. Let

$$\big(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, G, \hat{G}, e, X^{(1)}, \ldots, X^{(3q)}, \hat{X}^{(1)}, \ldots, \hat{X}^{(q+1)}\big)$$

be an instance of $(3q, q+1)$-DL, that is, for some x, we have $X^{(i)} = x^i G$ and $\hat{X}^{(i)} = x^i \hat{G}$. Reduction $\mathcal{B}$ chooses $c_1, \ldots, c_q \leftarrow\!\!\$\, \mathbb{Z}_p$; if for any $i \in [q] : -c_i G = X^{(1)}$ then $\mathcal{B}$ stops and returns $-c_i$.

Otherwise, $\mathcal{B}$ defines the polynomial

$$\prod_{j=1}^{q} (\mathrm{X} + c_j) = \sum_{j=0}^{q} \gamma_j \mathrm{X}^j =: \mathsf{P}(\mathrm{X})$$

for some $\gamma_0, \ldots, \gamma_q \in \mathbb{Z}_p$. It defines new generators $H := \sum_{j=0}^{q} \gamma_j X^{(j)} = \big(\prod_{j=1}^{q}(x + c_j)\big) G$ and $\hat{H} := \sum_{j=0}^{q} \gamma_j \hat{X}^{(j)}$. If $H = 0_1$ then $\mathcal{B}$ factors $\mathsf{P}(\mathrm{X})$ and returns the root x that satisfies $xG = X^{(1)}$. Since G and $\hat{G}$ were uniform, so are H and $\hat{H}$. The reduction then completes a q-PowDenDL challenge

$$\big(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, H, \hat{H}, e, Y^{(1)}, \ldots, Y^{(2q)}, \hat{Y}^{(1)}, Y_1, \ldots, Y_q, \hat{Y}_1, \ldots, \hat{Y}_q, c_1, \ldots c_1\big)$$

for secret x as follows:

- for $i \in [2q]$: $Y^{(i)} := \sum_{j=0}^{q} \gamma_j X^{(j+i)} = \big(x^i \prod_{j=1}^{q}(x + c_j)\big) G = x^i H$; and analogously $\hat{Y}^{(1)} = \big(x \prod_{j=1}^{q}(x + c_j)\big) \hat{G} = x\hat{H}$;
- for $i \in [q]$: let $\delta_{i,j}$ be such that $\sum_{j=0}^{q-1} \delta_{i,j} \mathrm{X}^j = \prod_{j=1, j\neq i}^{q} (\mathrm{X} + c_j)$; set $Y_i := \sum_{j=0}^{q-1} \delta_{i,j} X^{(j)} = \big(\prod_{j=1,j\neq i}^{q}(x + c_j)\big) G = \frac{1}{x+c_i} H$ and, likewise, $\hat{Y}_i := \sum_{j=0}^{q-1} \delta_{i,j} \hat{X}^{(j)}$.

Reduction $\mathcal{B}$ runs $\mathcal{A}$ on the (correctly distributed) q-PowDenDL instance and forwards the solution x if $\mathcal{A}$ finds it. Whenever $\mathcal{A}$ finds it, $\mathcal{B}$ also solves its $(3q, q+1)$-DL challenge.

Theorem 1. *Let $q \in \mathbb{N}$ and let $\mathcal{A}$ be an algebraic adversary attacking* $\mathrm{UNF}^{\mathrm{AGM}}$ *of FHS that makes q signing queries. Then there exists a reduction $\mathcal{B}$ against q-PowDenDL for* BGGen *such that*

$$\mathsf{Adv}^{q\text{-PowDenDL}}_{\mathsf{BGGen},\mathcal{B}} \geq \mathsf{Adv}^{\mathrm{UNF}^{\mathrm{AGM}}}_{\mathrm{FHS},\mathcal{A}} - \frac{4q+1}{p-1}.$$

Proof Idea. We will construct a reduction that essentially views the discrete logarithm z of any group element Z as a polynomial $\mathsf{Z}(\mathrm{Y})$ in indeterminate Y, such that when evaluated on the solution y of the given q-PowDenDL challenge, we have $\mathsf{Z}(y) = z$. In particular, the reduction embeds y into the public key $\hat{\boldsymbol{X}}$ given to the adversary, as well as into the randomness r_i that is sampled for the i-th signing query.

In order to guarantee independence of the adversary's behavior from y , we hide y by both multiplying with and adding a uniform element from $\mathbb{Z}_p$. In particular, components of the secret key will have the form $x_j y + x'_j$ for random x_j, x'_j. This ensures that even an unbounded adversary that can compute discrete logarithms is unable to reason about y, since it is information-theoretically hidden (the values x_j and x'_j are not used anywhere else). Using the element

$\hat{Y}^{(1)} = y\hat{G}$ from its q-PowDenDL instance, the reduction can compute the public key elements $\hat{X}_j = x_j\hat{Y}^{(1)} + x'_j\hat{G}$.

Analogously, y will be embedded into the randomness r_i of each signing query. We now show how the reduction answers its i-th signing query. Let $(Z_k, R_k, \hat{R}_k)$, $k < i$, be the signatures given to the adversary $\mathcal{A}$ so far, for which the reduction knows the polynomials representing their discrete logarithms. When $\mathcal{A}$ queries the signing oracle on a message $\boldsymbol{M}$, since $\mathcal{A}$ is algebraic, it accompanies each M_j with a representation $(\mu^{(j)}, \mu^{(j)}_{z,1}, \ldots, \mu^{(j)}_{z,i-1}, \mu^{(j)}_{r,1}, \ldots, \mu^{(j)}_{r,i-1})$ such that

$$M_j := \mu^{(j)}G + \sum_{k=1}^{i-1} \mu^{(j)}_{z,k} Z_k + \sum_{k=1}^{i-1} \mu^{(j)}_{r,k} R_k \ , \tag{6}$$

since $G, Z_1, \ldots, Z_{i-1}, R_1, \ldots, R_{i-1}$ are all the $\mathbb{G}_1$ elements that $\mathcal{A}$ has seen so far. As the reduction knows the polynomials associated to these group elements, from (6) it can compute the polynomial associated to M_j, and, from this, the polynomial associated to the signature element $Z = r\sum(x_i y + x'_i)M_i$. Even though the reduction does not know y, it can evaluate these polynomials on y "in the exponent" by performing group operations in $\mathbb{G}_1$ on the elements given in the q-PowDenDL challenge and thereby compute Z (and analogously R and $\hat{R}$).

Once the adversary submits its forgery $(Z, R, \hat{R}), M$, the reduction considers the two EQS verification equations (4) and (5) "in the exponent", and represents them in the "homogenious" form $e(R, \hat{G}) - e(G, \hat{R}) = 0$ (for (5)). The algebraic adversary $\mathcal{A}$ accompanies its forgery with representations, from which the reduction can compute the polynomials associated to each element. Plugging these into the verification equations, $\mathcal{B}$ computes two "verification polynomials" Q_1 and Q_2, which evaluate to 0 at y if and only if $\mathcal{A}$'s forgery satisfies the corresponding equation.

If the adversary succeeds, there are two cases: (1) At least one of the verification polynomials is not the zero polynomial: $\mathsf{Q}_i \not\equiv 0$. Then the q-PowDenDL solution y is a root of Q_i. By factoring Q_i, we therefore obtain the solution y. (2) Both polynomials are identically zero: we show that in this case the message on which the adversary provided a forgery was in fact a multiple of a previously asked query. This contradicts that the adversary wins the game. This will be accomplished by reasoning about the coefficients that the algebraic adversary provides by equating coefficients of the verification polynomial.

Proof. Consider the UNF$^{\mathrm{AGM}}$ game instantiated with FHS as shown in Fig. 2. (We omit the group elements from $\mathcal{A}$'s outputs, since they are determined by their representations.) We follow the convention that for an uppercase Latin letter A the coefficients will be represented by its greek lowercase analog α, where subscripts like $\alpha_{z,k}$ are to be read as "the coefficient that gets multiplied with Z_k". The elements Z_i, R_i and $\hat{R}_i$ are the answers to the i-th signing query. For example, the element Z is represented by the coefficients $\zeta, \zeta_{z,k}$ and $\zeta_{r,k}$ for $k \in [q]$, as can be seen in Fig. 2 on Line 7.

```
$\mathrm{UNF}^{\mathrm{AGM}}_{\mathrm{FHS},\mathcal{A}}(\lambda)$
1  $grp \leftarrow \mathsf{ParGen}(1^\lambda)$
2  $(sk, pk := \hat{\boldsymbol{X}}) \leftarrow \mathsf{KeyGen}(grp, 1^\ell)$
3  $Q := \emptyset$
4  $\Big(\big(\mu^{(j)}, (\mu^{(j)}_{z,k})_{k=1}^q, (\mu^{(j)}_{r,k})_{k=1}^q\big)_{j=1}^\ell,$
   $\big(\zeta, (\zeta_{z,k})_{k=1}^q, (\zeta_{r,k})_{k=1}^q\big), \big(\rho, (\rho_{z,k})_{k=1}^q, (\rho_{r,k})_{k=1}^q\big),$
   $\big(\hat\rho, (\hat\rho_{x,k})_{k=1}^\ell, (\hat\rho_{r,k})_{k=1}^q\big)\Big) \leftarrow \mathcal{A}^{\mathcal{O}}(grp, pk)$
5  For $j \in [\ell] : M^*_j := \mu^{(j)}G + \sum_{k=1}^q \mu^{(j)}_{z,k} Z_k + \sum_{k=1}^q \mu^{(j)}_{r,k} R_k$
6  if $\boldsymbol{M}^* \in Q$ : return $0$
7  $Z := \zeta G + \sum_{k=1}^q \zeta_{z,k} Z_k + \sum_{k=1}^q \zeta_{r,k} R_k$
8  $R := \rho G + \sum_{k=1}^q \rho_{z,k} Z_k + \sum_{k=1}^q \rho_{r,k} R_k$
9  $\hat R := \hat\rho \hat G + \sum_{k=1}^\ell \hat\rho_{x,k} \hat X_k + \sum_{k=1}^q \hat\rho_{r,k} \hat R_k$
10 return $\sum_j e(M_j, \hat X_j) = e(Z, \hat R) \wedge e(R, \hat G) = e(G, \hat R)$

$\mathcal{O}\Big(\big(\mu^{(i,j)}, (\mu^{(i,j)}_{z,k})_{k=1}^{i-1}, (\mu^{(i,j)}_{r,k})_{k=1}^{i-1}\big)_{j=1}^\ell\Big)$   // the $i$-th query
1  For $j \in [\ell] : M_j := \mu^{(i,j)}G + \sum_{k=1}^{i-1} \mu^{(i,j)}_{z,k} Z_k + \sum_{k=1}^{i-1} \mu^{(i,j)}_{r,k} R_k$
2  $Q := Q \cup [(M_j)_j]_\sim$
3  $r \leftarrow\$ \mathbb{Z}_p^*$
4  return $\big(Z_i := r \sum_j x_j M_j, R_i := \frac{1}{r} G, \hat R_i := \frac{1}{r} \hat G\big)$
```

Fig. 2. The game $\mathrm{UNF}^{\mathrm{AGM}}$ for the EQS scheme FHS.

We will construct a reduction $\mathcal{B}$ in Fig. 3 that breaks q-PowDenDL using an algebraic adversary $\mathcal{A}$ against $\mathrm{UNF}^{\mathrm{AGM}}$ that makes up to q queries to the signing oracle. The reduction works as follows: it gets the q-PowDenDL challenge

$$\big(Y^{(i)}\big)_{i=1}^{2q}, \hat Y^{(1)}, \big(Y_i, \hat Y_i, c_i\big)_{i=1}^q$$

with the aim of computing the discrete logarithm y of $Y^{(1)}$. For the sake of convenience define $Y^{(0)} := G$. Sampling uniform vectors $\boldsymbol{x}, \boldsymbol{x}'$ the reduction embeds y into the secret key by setting the public key elements $\hat X_j := x_j \hat Y^{(1)} + x'_j \hat G$.

If for any j we have $\hat{X}_j = 0_2$ then $\mathcal{B}$ stops and returns $y = -x_j^{-1}x'_j \mod p$. Note that the public key elements are distributed correctly, since the reduction $\mathcal{B}$ essentially implements rejection sampling. The secret key elements, which correspond to the discrete logarithm of the public key, are thus of the form $x_j y + x'_j$. The reduction can therefore, without knowing y, evaluate a polynomial at y "in the exponent" by using the elements of the q-PowDenDL challenge. These polynomials will be represented in sans-serif font, e.g. M_j, and their indeterminates in Roman font, e.g. Y.

The vectors $\boldsymbol{x}$ and $\boldsymbol{x}'$ are required so the adversary's behavior is independent of y. Even if it is unbounded and is able to obtain discrete logarithms of elements, since y is hidden information-theoretically, it cannot reason about y. Similarly, y is embedded in the randomness r that gets introduced during a signing query. The signing randomness will be of the form as $r_i(y + c_i)$ where r_i gets drawn uniformly and c_i is part of the q-PowDenDL challenge. Due to the randomness r appearing both as r and its reciprocal $\frac{1}{r}$, the reduction will in fact consider Laurent polynomials. The elements Y_i and $\hat{Y}_i$ from its challenge enable the reduction to also evaluate (very specific) Laurent polynomials "in the exponent" at y and compute corresponding group elements. Since y is embedded in both the secret key and the signature randomness, the reduction considers *multivariate* Laurent polynomials in indeterminates $\mathbf{X}$ and $\mathbf{R}$, which models the adversary's view more closely and simplifies our reasoning. Only at a later stage these multivariate Laurent polynomials will be transformed into univariate polynomials in Y.

When $\mathcal{A}$ makes the i-th signing query, for $j \in [\ell]$ the reduction will receive coefficients

$$\mu^{(i,j)}, \left(\mu_{z,k}^{(i,j)}\right)_{k=1}^{i-1}, \left(\mu_{r,k}^{(i,j)}\right)_{k=1}^{i-1}$$

that represent the j-th component of the queried message

$$M_j := \mu^{(i,j)}G + \sum_{k=1}^{i-1} \mu_{z,k}^{(i,j)} Z_k + \sum_{k=1}^{i-1} \mu_{r,k}^{(i,j)} R_k.$$

Since Z_k and R_k were the answers to previous signing queries, the reduction has Laurent polynomials that represent their respective logarithms. This fact will be used to find a Laurent polynomial that represents the logarithm of the answer to the i-th query Z_i. The following lemma will give a detailed description on how the reduction answers the adversaries signing queries.

Lemma 4. *There exist coefficients $a_k^{(i)}$, $k \in \{0, \dots, 2i\}$, and $b_k^{(i)}$, $k \in [i-1]$, such that the polynomial Z_i representing the signature element Z_i from the i-th signing query is of the form*

$$\mathsf{Z}_i(\mathrm{Y}) = \sum_{k=0}^{2i} a_k^{(i)} \mathrm{Y}^k + \sum_{k=1}^{i-1} b_k^{(i)} \frac{1}{r_k(\mathrm{Y} + c_k)}.$$

Moreover $\mathcal{B}$ can compute these coefficients efficiently.

```
$\mathcal{B}^{\mathcal{A}}\Big(grp, \big(Y^{(i)}\big)_{i=1}^{2q}, \hat{Y}^{(1)}, \big(Y_i, \hat{Y}_i, c_i\big)_{i=1}^{q}\Big)$
1   $r_1, \dots, r_q, x_1, \dots, x_\ell \leftarrow\$ \mathbb{Z}_p^*; x'_1, \dots, x'_\ell \leftarrow\$ \mathbb{Z}_p$
2   $pk := \Big(x_1 \hat{Y}^{(1)} + x'_1 \hat{G}, \dots, x_\ell \hat{Y}^{(1)} + x'_\ell \hat{G}\Big)$
3   if $\exists j$ s.t. $x_j \hat{Y}^{(1)} + x'_j \hat{G} = 0$ : return $y := -x_j^{-1} x'_j \mod p$
4   $\Big(\Big(\mu^{(j)}, \big(\mu_{z,k}^{(j)}\big)_{k=1}^{q}, \big(\mu_{r,k}^{(j)}\big)_{k=1}^{q}\Big)_{j=1}^{\ell}, \Big(\big(\zeta, (\zeta_{z,k})_{k=1}^{q}, (\zeta_{r,k})_{k=1}^{q}\big),$
        $\big(\rho, (\rho_{z,k})_{k=1}^{q}, (\rho_{r,k})_{k=1}^{q}\big), \big(\hat{\rho}, (\hat{\rho}_{x,k})_{k=1}^{\ell}, (\hat{\rho}_{r,k})_{k=1}^{q}\big)\Big)\Big) \leftarrow \mathcal{A}^{\texttt{OSign}}(grp, pk)$
    For $j \in [\ell]$ : $\mathsf{M}_j(\mathbf{X}, \mathbf{R}) := \mu^{(j)} + \sum_{k=1}^{q} \mu_{z,k}^{(j)} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}) + \sum_{k=1}^{q} \mu_{r,k}^{(j)} \mathrm{R}_k^{-1}$
5   $\mathsf{R}(\mathbf{X}, \mathbf{R}) := \rho + \sum_{k=1}^{q} \rho_{z,k} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}) + \sum_{k=1}^{q} \rho_{r,k} \mathrm{R}_k^{-1}$
6   $\hat{\mathsf{R}}(\mathbf{X}, \mathbf{R}) := \hat{\rho} + \sum_{k=1}^{\ell} \hat{\rho}_{x,k} \mathrm{X}_k + \sum_{k=1}^{q} \hat{\rho}_{r,k} \mathrm{R}_k^{-1}$
7   $\mathsf{V}_1 := \Big(\prod_{i=1}^{q} \mathrm{R_i}\Big) \Big(\mathsf{R}(\mathbf{X}, \mathbf{R}) - \hat{\mathsf{R}}(\mathbf{X}, \mathbf{R})\Big)$
8   $\mathsf{V}_2 := \Big(\prod_{i=1}^{q} \mathrm{R}_i^2\Big)\Big(\Big(\sum_{j=1}^{\ell} \mathrm{X_j} \mathsf{M_j}\Big) - \hat{\mathsf{R}}(\mathbf{X}, \mathbf{R})\Big(\zeta + \sum_{\mathrm{k=1}}^{\mathrm{q}} \zeta_{\mathrm{z,k}} \mathsf{Z}_{\mathrm{k}}(\mathbf{X}, \mathbf{R}) + \sum_{\mathrm{k=1}}^{\mathrm{q}} \zeta_{\mathrm{r,k}} \mathrm{R_k^{-1}}\Big)\Big)$
9   For $t \in [2]$ : $\mathsf{Q}_t := \mathsf{V}_t\Big(\big(x_i \mathrm{Y} + x'_i\big)_{i=1}^{\ell}, \big(r_i(\mathrm{Y} + c_i)\big)_{i=1}^{q}\Big)$
10  $S := \emptyset$;  if $\mathsf{Q}_1 \not\equiv 0$ : $S := S \cup \mathsf{Roots}(\mathsf{Q}_1)$;  if $\mathsf{Q}_2 \not\equiv 0$ : $S := S \cup \mathsf{Roots}(\mathsf{Q}_2)$
11  if $\exists y \in S$ s.t. $yG = Y^{(1)}$ : return $y$
```

```
$\texttt{OSign}\Big(\Big(\mu^{(i,j)}, \big(\mu_{z,k}^{(i,j)}\big)_{k=1}^{i-1}, \big(\mu_{r,k}^{(i,j)}\big)_{k=1}^{i-1}\Big)_{j=1}^{\ell}\Big)$   // describing the $i$-th signing query
1   For $j \in [\ell]$ :  $\mathsf{M}_j^{(i)}(\mathbf{X}, \mathbf{R}) := \mu^{(i,j)} + \sum_{k=1}^{i-1} \mu_{z,k}^{(i,j)} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}) + \sum_{k=1}^{i-1} \mu_{r,k}^{(i,j)} \mathrm{R}_k^{-1}$
2   $\mathsf{Z}_i(\mathbf{X}, \mathbf{R}) := \mathrm{R}_i \sum_{j=1}^{\ell} \mathrm{X}_j \mathsf{M}_j^{(i)}(\mathbf{X}, \mathbf{R})$
3   Parse $\mathsf{Z}_i\Big(\big(x_j \mathrm{Y} + x'_j\big)_{j=1}^{\ell}, \big(r_j(\mathrm{Y} + c_j)\big)_{j=1}^{i-1}\Big)$ as $\mathsf{Z}_i = \sum_{j=0}^{2i} a_j^{(i)} \mathrm{Y}^i + \sum_{j=1}^{i-1} b_j^{(i)} \frac{1}{r_j(\mathrm{Y} + c_j)}$
4   $Z_i := \sum_{j=0}^{2i} a_j^{(i)} Y^{(i)} + \sum_{j=1}^{i-1} b_j^{(i)} Y_j$
5   return $\Big(Z_i, R_i := \frac{1}{r_i} Y_i, \hat{R}_i := \frac{1}{r_i} \hat{Y}_i\Big)$
```

Fig. 3. Reduction from FHS unforgeability in the AGM to q-PowDenDL

Proof. We will prove this by induction on the signing queries. Consider $i = 1$, the first signing query. As the previously seen $\mathbb{G}_1$ element is G, for the message we have $M_j^{(1)} = \mu^{(1,j)} G$ for some $\mu^{(1,j)}$, which we represent as a polynomial

$\mathsf{M}_j^{(1)} = \mu^{(1,j)}$ for $j \in [\ell]$. Therefore the reduction will consider the Laurent polynomial

$$\mathsf{Z}_1(\mathbf{X}, \mathbf{R}) = \mathrm{R}_1 \sum_{j=1}^{\ell} \mu^{(1,j)} \mathrm{X}_j$$

evaluated on $\mathrm{X}_j = x_j \mathrm{Y} + x_j'$ for $j \in [\ell]$ and $\mathrm{R}_1 = r_1(\mathrm{Y} + c_1)$, which can be parsed as (recall that $\odot$ denotes componentwise multiplication)

$$\begin{aligned}\mathsf{Z}_1(\boldsymbol{x}\mathrm{Y} + \boldsymbol{x}', \boldsymbol{r}\mathrm{Y} + \boldsymbol{r} \odot \boldsymbol{c}) &= r_1(\mathrm{Y} + c_1) \sum_j \mu^{(1,j)} (x_j \mathrm{Y} + x_j') \\ &= \mathrm{Y}^2 \sum_j \mu^{(1,j)} r_1 x_j + \mathrm{Y} \sum_j \mu^{(1,j)} r_1 \left(x_j c_1 + x_j'\right) + \sum_j \mu^{(1,j)} r_1 c_1 x_j' \\ &= \sum_{k=0}^{2} a_k^{(1)} \mathrm{Y}^k,\end{aligned}$$

for appropriate coefficients $a_k^{(1)}$. Observe that $\deg_{\mathrm{Y}} \mathsf{Z}_1 \leq 2$. The reduction then sends the group elements $Z_1 := \sum_{k=0}^{2} a_k^{(1)} Y^{(k)}$ and $R_1 := \frac{1}{r_1} Y_1 = \frac{1}{r_1(y+c_1)} G$ and $\hat{R}_1 := \frac{1}{r_1} \hat{Y}_1$ answering the query. Note that the fractional part of Z_1 being zero is in accordance with the statement of this lemma, since $\sum_{k=1}^{0} b_k^{(1)} (r_k(\mathrm{Y} + c_k))^{-1} = 0$ holds for the empty sum.

Now consider the i-th query, after the reduction has answered all queries $k < i$ represented by polynomials Z_k for which $\deg_{\mathrm{Y}} \mathsf{Z}_k \leq 2k$ holds. The previously seen $\mathbb{G}_1$ elements additionally contain Z_k and R_k for $k < i$, therefore the message is provided with coefficients such that for $j \in [\ell]$ its polynomial representation is

$$\mathsf{M}_j^{(i)} = \mu^{(i,j)} + \sum_k^{i-1} \mu_{z,k}^{(i,j)} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}) + \sum_k^{i-1} \mu_{r,k}^{(i,j)} \mathrm{R}_k^{-1}.$$

The reduction then considers the Laurent polynomial

$$\mathsf{Z}_i(\mathbf{X}, \mathbf{R}) = \mathrm{R}_i \sum_j \mathrm{X}_j \left(\mu^{(i,j)} + \sum_k^{i-1} \mu_{z,k}^{(i,j)} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}) + \sum_k^{i-1} \mu_{r,k}^{(i,j)} \mathrm{R}_k^{-1} \right)$$

and evaluates it on $\mathrm{X}_j = x_j \mathrm{Y} + x_j'$ for $j \in [\ell]$ and $\mathrm{R}_i = r_i(\mathrm{Y} + c_i)$ for $i \in [q]$. By the induction hypothesis we know that for $k < i$ there exist coefficients $a_j^{(k)}$ and $b_j^{(k)}$ such that

$$\mathsf{Z}_k(\boldsymbol{x}\mathrm{Y} + \boldsymbol{x}', \boldsymbol{r}\mathrm{Y} + \boldsymbol{r} \odot \boldsymbol{c}) = \sum_{j=0}^{2k} a_j^{(k)} \mathrm{Y}^j + \sum_{j=1}^{k-1} b_j^{(k)} \frac{1}{r_j(\mathrm{Y} + c_j)}.$$

The reduction now considers $\mathsf{Z}_i(\boldsymbol{x}\mathrm{Y} + \boldsymbol{x}', \boldsymbol{r}\mathrm{Y} + \boldsymbol{r} \odot \boldsymbol{c}) = \mathsf{P}(\mathrm{Y}) + \mathsf{F}(\mathrm{Y})$ where P denotes the polynomial part, while F denotes the fractional part of Z_i. The polynomial part can be parsed as

$$\mathsf{P}(\mathrm{Y}) = r_i(\mathrm{Y}+c_i)\sum_j (x_j\mathrm{Y}+x'_j)\left(\mu^{(i,j)} + \sum_k^{i-1} \mu_{z,k}^{(i,j)} \sum_{j=0}^{2k} a_j^{(k)}\mathrm{Y}^j\right) = \sum_{j=0}^{2i} p_j \mathrm{Y}^j,$$

for appropriate coefficients p_j. For the fractional part

$$\mathsf{F}(\mathrm{Y}) = r_i(\mathrm{Y}+c_i)\sum_j (x_j\mathrm{Y}+x'_j)\left(\sum_k^{i-1} \mu_{z,k}^{(i,j)} \sum_m^{k-1} b_m^{(k)} \frac{1}{r_m(\mathrm{Y}+c_m)} + \sum_k^{i-1} \mu_{r,k}^{(i,j)} \frac{1}{r_k(\mathrm{Y}+c_k)}\right),$$

the reduction can find coefficients f_j for $j \in \{1, 0, \dots, -i+1\}$ via partial fraction decomposition such that

$$\mathsf{F}(\mathrm{Y}) = f_1\mathrm{Y} + f_0 + \sum_{j=1}^{i-1} f_{-j}\frac{1}{r_j(\mathrm{Y}+c_j)}.$$

Therefore

$$\begin{aligned}\mathsf{Z}_i(\boldsymbol{x}\mathrm{Y}+\boldsymbol{x}', \boldsymbol{r}\mathrm{Y}+\boldsymbol{r}\odot\boldsymbol{c}) &= \mathsf{P}(\mathrm{Y}) + \mathsf{F}(\mathrm{Y}) \\ &= \sum_{j=0}^{2i} p_j\mathrm{Y}^j + f_1\mathrm{Y} + f_0 + \sum_{j=1}^{i-1} f_{-j}\frac{1}{r_j(\mathrm{Y}+c_j)} \\ &= \sum_{j=0}^{2i} a_j^{(i)}\mathrm{Y}^j + \sum_{j=1}^{i-1} b_j^{(i)}\frac{1}{r_j(\mathrm{Y}+c_j)},\end{aligned}$$

for appropriate $a_j^{(i)}$, $b_j^{(i)}$. The reduction answers the signing query with $Z_i := \sum_{j=0}^{2i} a_j^{(i)} Y^{(j)} + \sum_{j=1}^{i-1} b_j^{(i)} Y_j$ and $R_i := \frac{1}{r_i}Y_i$ and $\hat{R}_i := \frac{1}{r_i}\hat{Y}_i$. Note that since

$$\begin{aligned}Z_i &= \sum_{j=0}^{2i} a_j^{(i)} Y^{(j)} + \sum_{j=1}^{i-1} b_j^{(i)} Y_j \\ &= \mathsf{Z}_i(\boldsymbol{x}y+\boldsymbol{x}', \boldsymbol{r}y+\boldsymbol{r}\odot\boldsymbol{c})G \\ &= (r_iy + r_ic_i)\sum_j M_j^{(i)}(x_jy+x'_j) \\ &= \tilde{r}\sum_j \tilde{x}_j M_j^{(i)},\end{aligned}$$

with $\tilde{r}$ being uniform in $\mathbb{Z}_p^*$, $\tilde{x}_j$ being consistent with $\hat{\boldsymbol{X}}$, and $R_i = \frac{1}{\tilde{r}}G$, the signatures are distributed identically to signatures from FHS. □

Since the q-PowDenDL challenge contains "powers" up to $2q$ and q different "denominators" we obtain the following corollary.

Corollary 1. *Using the q-PowDenDL challenge, the reduction can answer q queries to the signing oracle.*

The following observation directly follows from the definition of Z_i, and noting that there do not exist reciprocal terms in $\mathbf{X}$ in any of the Laurent polynomials that we consider.

Remark 1. Let $i \in [q]$. Then for every monomial m of Z_i there exists a $j \in [q]$ such that X_j divides m.

At some point the adversary $\mathcal{A}$ will output coefficients that represent a forgery consisting of a message $\boldsymbol{M}$ and the signature $(Z, R, \hat{R})$. Figure 2 describes how these coefficients relate to the elements. $\mathcal{B}$ then defines polynomials V_1 and V_2 that correspond to the two verification equations in the $\mathrm{UNF}^{\mathrm{AGM}}$ game. If the verification equation (5) $e(G, \hat{R}) = e(R, \hat{G})$ holds, then the logarithms of R and $\hat{R}$ are equivalent. Therefore the polynomial V_1 has y as a zero if R and $\hat{R}$ satisfy that verification equation of the game.

Recall the definitions of the Laurent polynomials (Fig. 3, lines 5 and 6):

$$\mathsf{R}(\mathbf{X}, \mathbf{R}) := \rho + \sum_{k=1}^{q} \rho_{z,k} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}) + \sum_{k=1}^{q} \rho_{r,k} \mathrm{R}_k^{-1}$$
$$\hat{\mathsf{R}}(\mathbf{X}, \mathbf{R}) := \hat{\rho} + \sum_{k=1}^{\ell} \hat{\rho}_{x,k} \mathrm{X}_k + \sum_{k=1}^{q} \hat{\rho}_{r,k} \mathrm{R}_k^{-1},$$

clearly, $\hat{\mathsf{R}}$ has denominators of maximum degree 1. Recall also that every Laurent polynomial we consider only has reciprocal terms in $\mathbf{R}$. Consider how Z_i is formed inductively, then Z_1 does not have any reciprocal terms (and insofar denominators with a maximum degree of 1), while when Z_i is formed from the previous Z_k for $k < i$, there might be reciprocal terms of degree 1 that are added. Therefore Z_i only has denominators with a maximum degree of 1, and so R only has denominators with a maximum degree of 1. Therefore, the factor $\prod_i \mathrm{R}_\mathrm{i}$ ensures that V_1 is a polynomial.

Analogously, V_2 has y as a root if and only if the verification equation (4) $\sum_j e(M_j, \hat{X}_j) = e(Z, \hat{R})$ holds. Since multiplying $\hat{\mathsf{R}}$ by R_k^{-1} potentially contained in the polynomial associated with Z creates denominators of degree 2, the factor $\prod_i \mathrm{R}_i^2$ ensures that V_2 is a polynomial. Observing that Z_k has a total degree upper-bounded by $2k$, the following corollary summarizes this argument.

Corollary 2. *Both V_1 and V_2 are polynomials of total degree upper-bounded by $4q + 1$.*

The following convention will simplify the remainder of the proof.

Remark 2. Since for fixed k the coefficient $\zeta_{z,k}$ only occurs as a factor of Z_k, whenever $\mathsf{Z}_k \equiv 0$ the adversary $\mathcal{A}$ can choose $\zeta_{z,k}$ arbitrarily. Since this choice does not change the system of equations, the reduction will set $\zeta_{z,k} := 0$ whenever $\mathsf{Z}_k \equiv 0$.

Note that this remark is not limited to $\zeta_{z,k}$ but also applies to other coefficients for example $\rho_{z,k}$ among others.

We will briefly discuss the technique used in the following proofs. Recall that if $\mathcal{R}$ is a ring, then an ideal $\mathfrak{I}$ is an additive subgroup of $\mathcal{R}$ such that for $\mathsf{P} \in \mathcal{R}$ and $\mathsf{Q} \in \mathfrak{I}$ it holds that $\mathsf{PQ} \in \mathfrak{I} \ni \mathsf{QP}$. For a subset S of $\mathcal{R}$, the ideal *generated by* S is defined as the smallest ideal $\mathfrak{I}$ such that $S \subseteq \mathfrak{I}$. If $\mathfrak{I}$ is an ideal then $\mathcal{R}/\mathfrak{I}$ is a ring, the so-called quotient ring or factor ring. Conceptually, the ideal "defines" which elements we identify with 0 in the quotient ring. Since we consider $\mathsf{V}_1 \equiv 0$, that is, for all inputs it vanishes, viewing this equation in the quotient ring corresponds to fixing specific terms (the ones in the ideal) to zero. This greatly simplifies notation when we equate coefficients of polynomials.

The following lemma states that given polynomials P_j in $\mathbf{X}$ such that (7) holds, then all P_j must vanish. Since we will apply this lemma to polynomials P_j of degree less than two, this means that they must be the zero polynomial. Equations of the form (7) emerge in the proof by considering $\mathsf{V}_t \equiv 0$, for $t \in [2]$, in an appropriate quotient ring. Remark 2 motivates that we merely need to consider the non-zero polynomials Z_j.

Lemma 5. *Let $J := \{j \mid \mathsf{Z}_j \not\equiv 0\} \subseteq [q]$ be the set of indices such that Z_j is a non-zero Laurent polynomial, and for $j \in J$ let $\mathsf{P}_j \in \mathbb{Z}_p[\mathbf{X}]$ be arbitrary polynomials. Then whenever*

$$\Big(\prod_k \mathrm{R}_k\Big) \sum_{j \in J} \mathsf{P}_j \mathsf{Z}_j \equiv 0, \tag{7}$$

as a polynomial in $\mathbf{X}$ and $\mathbf{R}$, we have $\mathsf{P}_j \equiv 0$ for all $j \in J$.

Proof. For $j \in J$ let $\mathfrak{K}_j$ be the ideal generated by $\{\mathrm{R}_i^2 \mid j < i \leq q\}$. We will consider equations in the factor rings $\mathbb{Z}_p[\mathbf{X}, \mathbf{R}]/\mathfrak{K}_j$, where we will denote equality by $\equiv_{\mathfrak{K}_j}$.

We will prove the claim inductively on the size of J. Assume $J \neq \emptyset$ and let $j := \min J$. Consider (7) modulo $\mathfrak{K}_j$. Since R_i divides Z_i, and thus R_i^2 divides $\big(\prod_k \mathrm{R}_k\big)\mathsf{Z}_i$, all the summands $\mathsf{P}_i\mathsf{Z}_i$ for $i > j$ vanish:

$$\Big(\prod_k \mathrm{R}_k\Big) \mathsf{P}_j \mathsf{Z}_j \equiv_{\mathfrak{K}_j} 0. \tag{8}$$

Now since $\mathsf{Z}_j \not\equiv 0$, and Z_j does not contain any R_i for $i > j$, we get

$$\deg_{\mathrm{R}_i} \Big(\prod_k \mathrm{R}_k\Big) \mathsf{Z}_j = 1.$$

Since P_j does not depend on $\mathbf{R}$, the only way that the left-hand side of (8) always vanishes is for $\mathsf{P}_j \equiv 0$. Considering $J' := J \setminus \{j\}$ we can inductively apply this reasoning eventually yielding the statement. □

We will now consider what it means if either verification polynomial V_t is the zero polynomial. This essentially models an adversary that tries to "outsmart" the reduction, by choosing coefficients in a way such that the polynomials V_1 and V_2 leak no information about y. In particular, if $\mathsf{V}_1 \equiv 0$ then $\mathcal{A}$ was (partly) successful in forging a signature but our reduction cannot obtain the q-PowDenDL solution y from the corresponding equation. The following lemma captures that the only way for $\mathcal{A}$ to enforce $\mathsf{V}_1 \equiv 0$ is to represent both R and $\hat{\mathsf{R}}$ by the same coefficients.

Lemma 6. *If* $\mathsf{V}_1 \equiv 0$*, then the following holds for the coefficients of* R *and* $\hat{\mathsf{R}}$*:*

$$\begin{aligned} \rho &= \hat{\rho}, \\ \rho_{z,j} &= \hat{\rho}_{x,j} = 0 \quad && \forall j \in [q], \\ \rho_{r,j} &= \hat{\rho}_{r,j} && \forall j \in [q]. \end{aligned}$$

Proof. Let $j \in [q]$ and let $\mathfrak{J}$ denote the ideal generated by $\{\mathrm{X}_1, \dots, \mathrm{X}_\ell, \mathrm{R}_j\}$. We will look at V_1 in the quotient ring $\mathbb{Z}_p[\mathbf{X}, \mathbf{R}]/\mathfrak{J}$. By $\equiv_{\mathfrak{J}}$ we denote equivalence in the quotient ring. Recall that

$$\begin{aligned} \mathsf{V}_1(\mathbf{X}, \mathbf{R}) &= \Big(\prod_i \mathrm{R}_i\Big)\Big(\mathsf{R}(\mathbf{X}, \mathbf{R}) - \hat{\mathsf{R}}(\mathbf{X}, \mathbf{R})\Big) \\ &= \Big(\prod_i \mathrm{R}_i\Big)\Big(\rho + \sum_k \rho_{z,k} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}) + \sum_k \rho_{r,k} \mathrm{R}_k^{-1} \\ &\qquad\qquad - \hat{\rho} - \sum_k \hat{\rho}_{x,k} \mathrm{X}_k - \sum_k \hat{\rho}_{r,k} \mathrm{R}_k^{-1}\Big). \end{aligned}$$

Now since $\rho \prod_i \mathrm{R}_i \equiv_{\mathfrak{J}} 0$ and $\hat{\rho} \prod_i \mathrm{R}_i \equiv_{\mathfrak{J}} 0$, and all monomials of Z_k contain some X_j (as noted in Remark 1), which implies $\mathsf{Z}_k = \mathrm{R}_k \sum_j \mathrm{X}_j \mathsf{M}_j^{(k)} \equiv_{\mathfrak{J}} 0$, we get

$$\begin{aligned} 0 \equiv \mathsf{V}_1(\mathbf{X}, \mathbf{R}) &\equiv_{\mathfrak{J}} \Big(\prod_i \mathrm{R}_i\Big)\Big(\sum_k (\rho_{r,k} - \hat{\rho}_{r,k}) \mathrm{R}_k^{-1}\Big) \\ &\equiv_{\mathfrak{J}} \Big(\prod_{i \neq j} \mathrm{R}_i\Big)(\rho_{r,j} - \hat{\rho}_{r,j}), \end{aligned}$$

where the second equivalence follows from $\mathrm{R}_j \equiv_{\mathfrak{J}} 0$. Equating coefficients yields $\rho_{r,j} = \hat{\rho}_{r,j}$. As j was arbitrary, this result holds for every $j \in [q]$.

Since $\mathsf{V}_1 \equiv 0$ implies that $\mathsf{V}_1 / \prod_i \mathrm{R}_i \equiv 0$ where it is defined, viewing this equation in the factor ring obtained by factoring the ideal $\mathfrak{X}$ generated by $\{\mathrm{X}_1, \dots, \mathrm{X}_\ell\}$ and using what we deduced about $\rho_{r,j}$ and $\hat{\rho}_{r,j}$ we get

$$\begin{aligned} \frac{\mathsf{V}_1(\mathbf{X}, \mathbf{R})}{\prod_i \mathrm{R}_i} &\equiv \rho - \hat{\rho} + \sum_k \rho_{z,k} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}) - \sum_k \hat{\rho}_{x,k} \mathrm{X}_k \\ &\equiv_{\mathfrak{X}} \rho - \hat{\rho}, \end{aligned}$$

where by Remark 1 we have $\mathsf{Z}_k = \mathrm{R}_k \sum_j \mathrm{X}_j \mathsf{M}_j^{(k)} \equiv_{\mathfrak{X}} 0$. We thus obtain $\rho = \hat{\rho}$.

Now consider the ideal $\mathfrak{R}$ generated by $\{\mathrm{R}_1, \ldots, \mathrm{R}_q\}$. Together with what we deduced so far we have

$$\begin{aligned} 0 \equiv \frac{\mathsf{V}_1(\mathbf{X}, \mathbf{R})}{\prod_i \mathrm{R}_i} &\equiv \sum_k \rho_{z,k} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}) - \sum_k \hat{\rho}_{x,k} \mathrm{X}_k \\ &\equiv \sum_k \rho_{z,k} \mathrm{R}_k \sum_j \mathrm{X}_j \mathsf{M}_j^{(k)}(\mathbf{X}, \mathbf{R}) - \sum_k \hat{\rho}_{x,k} \mathrm{X}_k \\ &\equiv_{\mathfrak{R}} - \sum_k \hat{\rho}_{x,k} \mathrm{X}_k, \end{aligned}$$

where we used that $\mathrm{M}_j^{(k)}$ only depends on R_i for $i < k$, and thus does not contain any inverses of R_k. By equating coefficients for X_k we obtain $\hat{\rho}_{x,k} = 0$ for all $k \in [\ell]$. We therefore showed that

$$\mathsf{V}_1(\mathbf{X}, \mathbf{R}) = \big(\prod_i \mathrm{R}_i\big) \sum_k \rho_{z,k} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}).$$

Analogously to Remark 2, the reduction can set $\rho_{z,k} = 0$ whenever $\mathsf{Z}_k \equiv 0$ without loss of generality. Thus, applying Lemma 5 yields $\rho_{z,j} = 0$ for $j \in [q]$. This concludes the proof. □

The next lemma captures that if both polynomials representing the verification equations are zero, then $\mathcal{A}$ must have provided a forgery on a message that is a multiple of a previously queried message. The idea here is to consider $\mathsf{V}_2 \equiv 0$ and iteratively compare coefficients in various quotient rings to simplify the equation such that we can reason about the coefficients provided by $\mathcal{A}$.

Lemma 7. *If $\mathsf{V}_1 \equiv 0$ and $\mathsf{V}_2 \equiv 0$, then the message returned by $\mathcal{A}$ is a multiple of a message queried to the signing oracle, in particular*

$$\exists i^* \in [q] \quad \forall j \in [\ell] : \quad M_j = \rho_{r,i^*} \zeta_{z,i^*} M_j^{(i^*)}.$$

Proof. By Lemma 6 we have

$$\hat{\mathsf{R}}(\mathbf{X}, \mathbf{R}) = \rho + \sum_k \rho_{r,k} \mathrm{R}_k^{-1},$$

and therefore V_2 has the form

$$\mathsf{V}_2(\mathbf{X}, \mathbf{R}) = \big(\prod_i \mathrm{R}_i^2\big) \Big(\sum_k \mathrm{X}_k \mathsf{M}_k(\mathbf{X}, \mathbf{R}) - \big(\rho + \sum_k \rho_{r,k} \mathrm{R}_k^{-1}\big) \cdot \mathsf{Z}(\mathbf{X}, \mathbf{R}) \Big).$$

Let $j \in [q]$ and denote by $\mathfrak{I}$ the ideal generated by $\{\mathrm{X}_1, \ldots, \mathrm{X}_\ell, \mathrm{R}_j\}$. Recall

$$\mathsf{Z}(\mathbf{X}, \mathbf{R}) = \zeta + \sum_i \zeta_{z,i} \mathsf{Z}_i(\mathbf{X}, \mathbf{R}) + \sum_i \zeta_{r,i} \mathrm{R}_i^{-1}.$$

Viewing V_2 modulo $\mathfrak{I}$, all the terms containing X_k vanish (cf. Remark 1), and the only terms that remain are the ones where R_j^2 cancels. We obtain

$$0 \equiv \mathsf{V}_2 \equiv_{\mathfrak{I}} -\rho_{r,j}\zeta_{r,j}, \tag{9}$$

where j was arbitrary. We will start by showing that $\zeta_{r,k} = 0$ for all $k \in [q]$. Assume towards a contradiction that there exists $k \in [q]$ such that $\zeta_{r,k} \neq 0$. Using this consider $j \neq k \in [q]$ and let $\mathfrak{J}$ denote the ideal generated by $\{\mathrm{X}_1, \ldots, \mathrm{X}_\ell, \mathrm{R}_k, \mathrm{R}_j\}$. Then, analogously to the previous step where we considered the ideal $\mathfrak{I}$, all the terms containing X_k vanish, and the only terms that remain are ones that don't contain R_k or R_j either:

$$\begin{aligned} 0 \equiv \frac{-\mathsf{V}_2}{\prod_i \mathrm{R}_i} &\equiv_{\mathfrak{J}} \Big(\prod_i \mathrm{R}_i\Big)\Big(\sum_m \sum_{m' \neq m} \rho_{r,m}\zeta_{r,m'}\mathrm{R}_m^{-1}\mathrm{R}_{m'}^{-1}\Big) \\ &\equiv_{\mathfrak{J}} \Big(\prod_{i \neq k, i \neq j} \mathrm{R}_i\Big)\Big(\rho_{r,k}\zeta_{r,j} + \rho_{r,j}\zeta_{r,k}\Big), \end{aligned}$$

and thus

$$\rho_{r,k}\zeta_{r,j} + \rho_{r,j}\zeta_{r,k} = 0 \quad \text{for all} \quad j \in [q]. \tag{9a}$$

Now since we assumed $\zeta_{r,k} \neq 0$ from (9) we get $\rho_{r,k} = 0$, which with (9a) yields $\rho_{r,j}\zeta_{r,k} = 0$ and thus $\rho_{r,j} = 0$ for all $j \in [q]$. Again let $k \in [q]$ and denote by $\mathfrak{J}$ the ideal generated by $\{\mathrm{X}_1, \ldots, \mathrm{X}_\ell, \mathrm{R}_k\}$. We have that

$$0 \equiv \frac{\mathsf{V}_2(\mathbf{X}, \mathbf{R})}{\prod_i \mathrm{R}_i} \equiv_{\mathfrak{J}} \Big(\prod_i \mathrm{R}_i\Big)\Big(-\rho\big(\zeta + \sum_j \zeta_{r,j}\mathrm{R}_j^{-1}\big)\Big) \equiv_{\mathfrak{J}} -\rho\zeta_{r,k}\prod_{i \neq k} \mathrm{R}_i,$$

and therefore $\rho = 0$. With $\rho_{r,j} = 0$ for all $j \in [q]$, Lemma 6 now implies $\mathsf{R} \equiv 0$, which contradicts $\mathcal{A}$ providing a valid forgery. Therefore $\zeta_{r,k} = 0$ for $k \in [q]$, and thus

$$\mathsf{Z}(\mathbf{X}, \mathbf{R}) = \zeta + \sum_k \zeta_{z,k}\mathsf{Z}_k(\mathbf{X}, \mathbf{R}). \tag{10}$$

Denote by $\mathfrak{L}$ the ideal generated by $\{\mathrm{X}_1, \ldots, \mathrm{X}_\ell\}$. Then, by Remark 1, we have $\mathsf{Z}(\mathbf{X}, \mathbf{R}) \equiv_{\mathfrak{L}} \zeta$ and therefore

$$0 \equiv \frac{-\mathsf{V}_2(\mathbf{X}, \mathbf{R})}{\prod_i \mathrm{R}_i} \equiv_{\mathfrak{L}} \big(\prod_i \mathrm{R}_i\big)\big(\zeta\rho + \sum_j \zeta\rho_{r,i}\mathrm{R}_j^{-1}\big) \equiv_{\mathfrak{L}} \big(\prod_i \mathrm{R}_i\big)\zeta\hat{\mathsf{R}}(\mathbf{X}, \mathbf{R}).$$

Since both $\hat{\mathsf{R}} \not\equiv 0$ and $\prod_k \mathrm{R}_k \not\equiv 0$ modulo $\mathfrak{L}$, we get $\zeta = 0$. We therefore showed that (10) has the form

$$\mathsf{Z}(\mathbf{X}, \mathbf{R}) = \sum_k \zeta_{z,k}\mathsf{Z}_k(\mathbf{X}, \mathbf{R}).$$

We will now show that merely one summand is non-zero. Define $i^* := \max\{i \mid \zeta_{z,i} \neq 0\}$, and note that by Remark 2 we have $\mathsf{Z}_{i^*} \not\equiv 0$. Recall that we deduced

$$\mathsf{V}_2(\mathbf{X}, \mathbf{R}) = \big(\prod_i \mathrm{R}_i^2\big)\Big(\sum_k \mathsf{M}_k(\mathbf{X}, \mathbf{R})\mathrm{X}_k - \big(\rho + \sum_k \rho_{r,k}\mathrm{R}_k^{-1}\big)\sum_k^{i^*} \zeta_{z,k}\mathsf{Z}_k(\mathbf{X}, \mathbf{R})\Big)$$

and that M_k is defined as

$$\mathsf{M}_k(\mathbf{X}, \mathbf{R}) := \mu^{(k)} + \sum_j \mu_{z,j}^{(k)}\mathsf{Z}_j(\mathbf{X}, \mathbf{R}) + \sum_j \mu_{r,j}^{(k)}\mathrm{R}_j^{-1}. \tag{11}$$

We will show that $\mu_{z,j}^{(k)} = 0$ for $j > i^*$. Let $k^* := \sup\{k \mid \exists j : \mu_{z,k}^{(j)} \neq 0\}$. Again, similarly to Remark 2, we have $\mathsf{Z}_{k^*} \not\equiv 0$. Suppose $k^* > i^*$ and consider all the monomials of V_2 that are divisible by $\mathrm{R}_{k^*}^3$, that is, all the terms in which Z_{k^*} appears

$$\big(\prod_i \mathrm{R}_i^2\big)\Big(\sum_j \mu_{z,k^*}^{(j)}\mathsf{Z}_{k^*}(\mathbf{X}, \mathbf{R})\mathrm{X}_j\Big) \equiv 0.$$

Since $\mathsf{Z}_{k^*} \not\equiv 0$, equating coefficients we obtain $\mu_{z,k^*}^{(j)} = 0$ for all j, therefore $k^* \leq i^*$.

Now consider all the monomials of V_2 that are divisible by $\mathrm{R}_{i^*}^3$, that is, all the terms in which Z_{i^*} appears, and equate coefficients with the zero polynomial:

$$\mathsf{Z}_{i^*}(\mathbf{X}, \mathbf{R})\big(\prod_i \mathrm{R}_i^2\big)\Big(\sum_j \mu_{z,i^*}^{(j)}\mathrm{X}_j - \big(\rho + \sum_{k \neq i^*} \rho_{r,k}\mathrm{R}_k^{-1}\big)\zeta_{z,i^*}\Big) \equiv 0.$$

Since $\mathsf{Z}_{i^*} \not\equiv 0$, we can equate coefficients of X_j to obtain $\mu_{z,i^*}^{(j)} = 0$ for $j \in [\ell]$. This leaves us with the subtrahend, where due to $\zeta_{z,i^*} \neq 0$ equating coefficients yields $\rho = 0$ and $\rho_{r,k} = 0$ for $k \in [q] \setminus \{i^*\}$. Now since $\mathsf{R} \not\equiv 0$, we have $\rho_{r,i^*} \neq 0$. This leaves us with

$$\mathsf{V}_2(\mathbf{X}, \mathbf{R}) = \big(\prod_i \mathrm{R}_i^2\big)\Big(\sum_k \mathsf{M}_k(\mathbf{X}, \mathbf{R})\mathrm{X}_k - \rho_{r,i^*}\mathrm{R}_{i^*}^{-1}\sum_k^{i^*} \zeta_{z,k}\mathsf{Z}_k(\mathbf{X}, \mathbf{R})\Big), \tag{12}$$

and (11) becomes

$$\mathsf{M}_k(\mathbf{X}, \mathbf{R}) = \mu^{(k)} + \sum_j^{i^*-1} \mu_{z,j}^{(k)}\mathsf{Z}_k(\mathbf{X}, \mathbf{R}) + \sum_j \mu_{r,j}^{(k)}\mathrm{R}_j^{-1}.$$

Now consider the ideal $\mathfrak{I}$ generated by $\{\mathrm{R}_1^2, \ldots, \mathrm{R}_{i^*-1}^2, \mathrm{R}_{i^*}\}$. Recall that $\mathsf{Z}_k := \mathrm{R}_k \sum_j \mathsf{M}_j^{(k)}\mathrm{X}_j$, and that for $j \in [\ell]$ the Laurent polynomial $\mathsf{M}_j^{(k)}$ only has reciprocal terms in $\mathrm{R}_{k'}$ for $k > k'$. Then in this ideal the subtrahend of (12) vanishes, and the only remaining terms are those where R_{i^*} cancels:

$$0 \equiv \frac{\mathsf{V}_2(\mathbf{X}, \mathbf{R})}{\prod_k \mathrm{R}_k} \equiv_{\mathfrak{I}} \Big(\prod_{k \neq i^*} \mathrm{R}_k\Big)\sum_j \mu_{r,i^*}^{(j)}\mathrm{X}_j.$$

Equating coefficients for X_j, we obtain $\mu_{r,i^*}^{(j)} = 0$ for $j \in [\ell]$.

Consider the ideal $\mathfrak{J}$ generated by R_{i^*}. Then in the corresponding factor ring the non-zero terms will be those where R_{i^*} cancels. Since we just showed that M_k does not contain any inverses of R_{i^*}, this can only happen in the subtrahend of (12) and thus

$$0 \equiv \frac{-\mathsf{V}_2(\mathbf{X}, \mathbf{R})}{\prod_i \mathrm{R}_i} \equiv_{\mathfrak{J}} \big(\prod_{i \neq i^*} \mathrm{R}_i\big) \rho_{r,i^*} \sum_{k}^{i^*-1} \zeta_{z,k} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}).$$

Since $\rho_{r,i^*} \neq 0$ and the right-hand side is constant in R_{i^*}, multiplying by R_{i^*} yields

$$\big(\prod_k \mathrm{R}_k\big) \sum_{j}^{i^*-1} \zeta_{z,j} \mathsf{Z}_j(\mathbf{X}, \mathbf{R}) \equiv 0.$$

Now by Remark 2, Lemma 5 applies, and therefore $\zeta_{z,j} = 0$ for $j < i^*$. We therefore showed that there is exactly one index i^* such that $\zeta_{z,i^*} \neq 0$.

Now by expanding the representation of M_k, (12) simplifies to

$$\mathsf{V}_2(\mathbf{X}, \mathbf{R}) = \Big(\prod_i \mathrm{R}_i^2\Big) \bigg(\sum_k \Big(\mu^{(k)} + \sum_{j}^{i^*-1} \mu_{z,j}^{(k)} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}) + \sum_{j \neq i^*} \mu_{r,j}^{(k)} \mathrm{R}_j^{-1} \Big) \mathrm{X}_k \\ - \rho_{r,i^*} \mathrm{R}_{i^*}^{-1} \zeta_{z,i^*} \mathsf{Z}_{i^*}(\mathbf{X}, \mathbf{R}) \bigg). \quad (13)$$

For $j > i^*$ consider this equation in the factor ring obtained by factoring the ideal $\mathfrak{J}$ generated by R_j^2. Then since for $k < j$ we have that $\deg_{\mathrm{R}_j} \mathsf{Z}_k = 0$, in this factor ring the only terms remaining are those of (13) that contain R_j^{-1}. We get

$$0 \equiv \mathsf{V}_2(\mathbf{X}, \mathbf{R}) \equiv_{\mathfrak{J}} \mathrm{R}_j \big(\prod_{i \neq j} \mathrm{R}_i^2\big) \sum_k \mu_{r,j}^{(k)} \mathrm{X}_k,$$

and equating coefficients yields $\mu_{r,j}^{(k)} = 0$ for $j > i^*$.

Denote by $\mathfrak{I}$ the ideal generated by $\{\mathrm{X}_i \mathrm{X}_j \mid 1 \leq i \leq j \leq \ell\}$. Recall the definition

$$\mathsf{Z}_{i^*}(\mathbf{X}, \mathbf{R}) := \mathrm{R}_{i^*} \sum_j \bigg(\mu^{(i^*,j)} + \sum_{k}^{i^*-1} \mu_{z,k}^{(i^*,j)} \mathsf{Z}_k(\mathbf{X}, \mathbf{R}) + \sum_{k}^{i^*-1} \mu_{r,k}^{(i^*,j)} \mathrm{R}_k^{-1} \bigg) \mathrm{X}_j.$$

Consider (13)$\equiv 0$ in the corresponding factor ring, where by Remark 1 the terms containing $\mathsf{Z}_k \mathrm{X}_j$ vanish:

$$\begin{aligned}0 \equiv \mathsf{V}_2(\mathbf{X}, \mathbf{R}) \equiv_{\mathfrak{J}} & \left(\prod_i \mathrm{R}_i^2\right)\Bigg(\sum_j \left(\mu^{(j)} + \sum_k^{i^*-1} \mu_{r,k}^{(j)} \mathrm{R}_k^{-1}\right)\mathrm{X}_j \\ & \qquad - \rho_{r,i^*} \mathrm{R}_{i^*}^{-1} \zeta_{z,i^*} \sum_j \left(\mu^{(i^*,j)} + \sum_k^{i^*-1} \mu_{r,k}^{(i^*,j)} \mathrm{R}_k^{-1}\right)\mathrm{X}_j\Bigg) \\ \equiv_{\mathfrak{J}} & \left(\prod_i \mathrm{R}_i^2\right) \sum_j \mathrm{X}_j \Bigg(\mu^{(j)} - \rho_{r,i^*}\zeta_{z,i^*}\mu^{(i^*,j)} \\ & \qquad + \sum_k^{i^*-1} \left(\mu_{r,k}^{(j)} - \rho_{r,i^*}\zeta_{z,i^*}\mu_{r,k}^{(i^*,j)}\right) \mathrm{R}_k^{-1}\Bigg).\end{aligned}$$

Equating coefficients yields $\mu_{r,k}^{(j)} = \rho_{r,i^*}\zeta_{z,i^*}\mu_{r,k}^{(i^*,j)}$ for $j \in [\ell]$ and $k < i^*$, and $\mu^{(j)} = \rho_{r,i^*}\zeta_{z,i^*}\mu^{(i^*,j)}$ for $j \in [\ell]$. Therefore (13) simplifies to

$$\mathsf{V}_2(\mathbf{X}, \mathbf{R}) = \left(\prod_i \mathrm{R}_i^2\right)\left(\sum_k^{i^*-1} \left(\sum_j \left(\mu_{z,k}^{(j)} - \rho_{r,i^*}\zeta_{z,i^*}\mu_{z,k}^{(i^*,j)}\right) \mathrm{X}_j\right)\mathsf{Z}_k\right).$$

For $k < i^*$ define the polynomial

$$\mathsf{P}_k(\mathbf{X}) := \sum_j \left(\mu_{z,k}^{(j)} - \rho_{r,i^*}\zeta_{z,i^*}\mu_{z,k}^{(i^*,j)}\right) \mathrm{X}_j.$$

Whenever $\mathsf{Z}_k = 0$, similarly to Remark 2, we can suppose that $\mu_{z,k}^{(j)} = 0$ and $\mu_{z,k}^{(i^*,j)} = 0$ for all $j \in [\ell]$, which implies $\mathsf{P}_k = 0$. Lemma 5 is therefore applicable, yielding $\mathsf{P}_k \equiv 0$ for $k < i^*$. Equating coefficients yields $\mu_{z,k}^{(j)} = \rho_{r,i^*}\zeta_{z,i^*}\mu_{z,k}^{(i^*,j)}$ for $j \in [\ell]$ and $k < i^*$. Thus, for all $j \in [\ell]$ we derived

$$\begin{aligned}M_j &= \mu^{(j)} G + \sum_k^{i^*-1} \mu_{z,k}^{(j)} Z_j + \sum_k^{i^*-1} \mu_{r,k}^{(j)} R_k \\ &= \rho_{r,i^*}\zeta_{z,i^*}\mu^{(i^*,j)} G + \sum_k^{i^*-1} \rho_{r,i^*}\zeta_{z,i^*}\mu_{z,k}^{(i^*,j)} Z_j + \sum_k^{i^*-1} \rho_{r,i^*}\zeta_{z,i^*}\mu_{r,k}^{(i^*,j)} R_k \\ &= \rho_{r,i^*}\zeta_{z,i^*}\left(\mu^{(i^*,j)} G + \sum_k^{i^*-1} \mu_{z,k}^{(i^*,j)} Z_j + \sum_k^{i^*-1} \mu_{r,k}^{(i^*,j)} R_k\right) \\ &= \rho_{r,i^*}\zeta_{z,i^*} M_j^{i^*}.\end{aligned}$$

□

So far, we reasoned about the multivariate verification polynomials V_1 and V_2 where each indeterminate corresponds to one secret value that gets embedded.

$\mathcal{B}$ transforms these multivariate verification polynomials into univariate polynomials in Y by evaluating the indeterminates by specifying how y was embedded. This yields the univariate polynomials Q_1 and Q_2 with indeterminate Y.

This transformation from multivariate to univariate polynomials might turn a non-zero polynomial into the zero polynomial. The following lemma will show that in our specific setting this is unlikely to occur.

Lemma 8. *Conditioned on $\mathcal{A}$ winning* $\mathrm{UNF}^{\mathrm{AGM}}$ *we have that the probability that one of the univariate polynomials* Q_1 *and* Q_2 *is non-zero is overwhelming:*

$$\Pr\left[\mathsf{Q}_1 \not\equiv 0 \vee \mathsf{Q}_2 \not\equiv 0 \,\middle|\, \mathrm{UNF}_{\mathcal{A}}^{\mathrm{AGM}} = 1\right] \geq 1 - \frac{4q+1}{p-1}.$$

Proof. Assuming that $\mathcal{A}$ wins $\mathrm{UNF}^{\mathrm{AGM}}$, Lemma 7 yields $\mathsf{V}_1 \not\equiv 0 \vee \mathsf{V}_2 \not\equiv 0$. Assume the case $\mathsf{V}_2 \not\equiv 0$. From Corollary 2 we have that V_2 is a polynomial of total degree upper-bounded by $4q+1$. We can therefore apply the BFL Lemma (Lemma 2) to conclude that given $\mathsf{V}_2 \not\equiv 0$, the leading coefficient in Y of

$$\mathsf{V}_2'\big(\mathbf{U}, \mathbf{V}, \mathbf{U}', \mathbf{V}', \mathrm{Y}\big) := \mathsf{V}_2\big(\mathbf{U}\mathrm{Y} + \mathbf{U}', \mathbf{V}\mathrm{Y} + \mathbf{V}'\big)$$

is a polynomial in indeterminates $\mathbf{U}, \mathbf{V}$ of degree upper-bounded by $4q+1$. Call this polynomial $\mathsf{V}_{2,\max}' \in \mathbb{Z}_p[\mathbf{U}, \mathbf{V}]$. Recall that for $i \in [q]$, r_i and c_i are drawn uniformly from $\mathbb{Z}_p^*$ and $\mathbb{Z}_p$, respectively. Let $\mathbf{r}' := \mathbf{r} \odot \mathbf{c}$ and note that

$$\mathsf{Q}_2(\mathrm{Y}) = \mathsf{V}_2'\left(\mathbf{x}, \mathbf{r}, \mathbf{x}', \mathbf{r}', \mathrm{Y}\right). \tag{14}$$

Then by (14), it suffices to show $\mathsf{V}_{2,\max}' \not\equiv 0$ with overwhelming probability. Since $(\mathbb{Z}_p, +)$ and $(\mathbb{Z}_p^*, \cdot)$ are both cyclic, r_i' is uniformly distributed over $\mathbb{Z}_p$. Moreover, (r_i, r_i') is uniformly distributed over $\mathbb{Z}_p^* \times \mathbb{Z}_p$ since the function

$$\begin{aligned} f \colon \mathbb{Z}_p^* \times \mathbb{Z}_p &\to \mathbb{Z}_p^* \times \mathbb{Z}_p \\ (x, y) &\mapsto (x, x \cdot y) \end{aligned}$$

is a bijection. This implies that the vector $\mathbf{r}' = (\mathbf{r} \odot \mathbf{c})$ is independent from the vector $\mathbf{r}$, and therefore $(\mathbf{x}', \mathbf{r}')$ is independent of $(\mathbf{x}, \mathbf{r})$.

Furthermore, since $(\mathbf{x}, \mathbf{r})$ is completely hidden from the adversary's view (due to the additive mask $(\mathbf{x}', \mathbf{r}')$ added after the multiplication), it is independent from the coefficients of V_2' (determined by $\mathcal{A}$), and thus independent of the coefficients of $\mathsf{V}_{2,\max}$ (still fully determined by $\mathcal{A}$). Thus the Schwartz-Zippel Lemma (Lemma 1) applied to $\mathsf{V}_{2,\max}$ on uniform inputs $\boldsymbol{x}, \boldsymbol{r}$ yields

$$\Pr\left[\mathsf{V}_{2,\max} \not\equiv 0 \,\middle|\, \mathrm{UNF}_{\mathcal{A}}^{\mathrm{AGM}} = 1\right] \geq 1 - \frac{4q+1}{p-1}.$$

Overall, we get

$$\begin{aligned} \Pr\left[\mathsf{Q}_1 \not\equiv 0 \vee \mathsf{Q}_2 \not\equiv 0 \,\middle|\, \mathrm{UNF}_{\mathcal{A}}^{\mathrm{AGM}} = 1\right] &\geq \Pr\left[\mathsf{Q}_2 \not\equiv 0 \,\middle|\, \mathrm{UNF}_{\mathcal{A}}^{\mathrm{AGM}} = 1\right] \\ &= \Pr\left[\mathsf{V}_{2,\max} \not\equiv 0 \,\middle|\, \mathrm{UNF}_{\mathcal{A}}^{\mathrm{AGM}} = 1\right] \\ &\geq 1 - \frac{4q+1}{p-1}. \end{aligned}$$

Noticing that the case $\mathsf{V}_1 \not\equiv 0$ follows analogously concludes the proof. $\square$

We will proceed with the proof of Theorem 1. We will show that reduction $\mathcal{B}$'s advantage is close to $\mathcal{A}$'s advantage.

We know that $\mathcal{B}$ wins if $\mathcal{A}$ wins and $(\mathsf{Q}_1 \not\equiv 0$ or $\mathsf{Q}_2 \not\equiv 0)$, since $\mathcal{A}$ winning implies $\mathsf{Verify}(pk, \boldsymbol{M}, \sigma) = 1$, which means $e\big(R^*, G_2\big) = e\big(G, \hat{R}^*\big)$ and $\sum_{i=1}^{q} e\big(M_i^*, pk_i\big) = e\big(Z, \hat{R}^*\big)$. Therefore the logarithm y of $Y^{(1)}$ is a root of both Q_1 and Q_2. Since at least one of them is not identically zero, with its degree upper-bounded by $4q+1$, reduction $\mathcal{B}$ can efficiently factor the non-zero one to obtain y among its roots, and therefore solve q-PowDenDL. This yields

$$\begin{aligned}\mathsf{Adv}^{q\text{-PowDenDL}}_{\mathsf{BGGen},\mathcal{B}} &\geq \Pr\left[\mathrm{UNF}^{\mathrm{AGM}}_{\mathcal{A}} = 1 \wedge (\mathsf{Q}_1 \not\equiv 0 \vee \mathsf{Q}_2 \not\equiv 0)\right] \\ &= \Pr\left[\mathrm{UNF}^{\mathrm{AGM}}_{\mathcal{A}} = 1\right] \Pr\left[\mathsf{Q}_1 \not\equiv 0 \vee \mathsf{Q}_2 \not\equiv 0 \,\middle|\, \mathrm{UNF}^{\mathrm{AGM}}_{\mathcal{A}} = 1\right],\end{aligned}$$

and applying Lemma 8 yields

$$\geq \mathsf{Adv}^{\mathrm{UNF}^{\mathrm{AGM}}}_{\mathcal{A}} \left(1 - \frac{4q+1}{p-1}\right) \geq \mathsf{Adv}^{\mathrm{UNF}^{\mathrm{AGM}}}_{\mathcal{A}} - \frac{4q+1}{p-1}.$$

This concludes the proof of Theorem 1. □

Acknowledgments. The authors would like to thank the anonymous reviewers for ASIACRYPT 2024 for their very valuable comments. This work was funded by the Vienna Science and Technology Fund (WWTF) [10.47379/VRG18002] and by the Austrian Science Fund (FWF) [10.55776/F8515-N].

References

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010.

[AGHO11] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 649–666. Springer, Heidelberg, August 2011.

[AGO11] Masayuki Abe, Jens Groth, and Miyako Ohkubo. Separating short structure-preserving signatures from non-interactive assumptions. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 628–646. Springer, Heidelberg, December 2011.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, Heidelberg, May 2004.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 440–456. Springer, Heidelberg, May 2005.

[BCC+09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, Heidelberg, August 2009.

[BEK+20] Jan Bobolz, Fabian Eidens, Stephan Krenn, Daniel Slamanig, and Christoph Striecks. Privacy-preserving incentive systems with highly efficient point-collection. In Hung-Min Sun, Shiuh-Pyng Shieh, Guofei Gu, and Giuseppe Ateniese, editors, *ASIACCS 20*, pages 319–333. ACM Press, October 2020.

[BFL20] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 121–151. Springer, Heidelberg, August 2020.

[BFP21] Balthazar Bauer, Georg Fuchsbauer, and Antoine Plouviez. The one-more discrete logarithm assumption in the generic group model. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 587–617. Springer, Heidelberg, December 2021.

[BFR24] Balthazar Bauer, Georg Fuchsbauer, and Fabian Regen. On proving equivalence class signatures secure from non-interactive assumptions. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part I*, volume 14601 of *LNCS*, pages 3–36. Springer, Heidelberg, April 2024.

[BHKS18] Michael Backes, Lucjan Hanzlik, Kamil Kluczniak, and Jonas Schneider.Signatures with flexible public key: Introducing equivalence classes for public keys. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 405–434. Springer, Heidelberg, December 2018.

[BLL+19] Xavier Bultel, Pascal Lafourcade, Russell W. F. Lai, Giulio Malavolta, Dominique Schröder, and Sri Aravinda Krishnan Thyagarajan. Efficient invisible and unlinkable sanitizable signatures. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part I*, volume 11442 of *LNCS*, pages 159–189. Springer, Heidelberg, April 2019.

[BLS04] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. On the selection of pairing-friendly groups. In Mitsuru Matsui and Robert J. Zuccherato, editors, *SAC 2003*, volume 3006 of *LNCS*, pages 17–25. Springer, Heidelberg, August 2004.

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko.The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.

[Bow17] Sean Bowe. BLS12-381: New zk-SNARK elliptic curve construction, 2017. https://electriccoin.co/blog/new-snark-curve/.

[BRS23] Fabrice Benhamouda, Mariana Raykova, and Karn Seth. Anonymous counting tokens. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part II*, volume 14439 of *LNCS*, pages 245–278. Springer, Heidelberg, December 2023.

[BSW23] Christian Badertscher, Mahdi Sedaghat, and Hendrik Waldner.Unlinkable policy-compliant signatures for compliant and decentralized anonymous payments.Cryptology ePrint Archive, Paper 2023/1070, 2023. https://eprint.iacr.org/2023/1070.

[CDLP22] Aisling Connolly, Jérôme Deschamps, Pascal Lafourcade, and Octavio Perez-Kempner. Protego: Efficient, revocable and auditable anonymous credentials with applications to Hyperledger Fabric. In Takanori Isobe and Santanu Sarkar, editors, *INDOCRYPT 2022*, volume 13774 of *LNCS*, pages 249–271. Springer, 2022.

[CFF+21] Matteo Campanelli, Antonio Faonio, Dario Fiore, Anaïs Querol, and Hadrián Rodríguez. Lunar: A toolbox for more efficient universal and updatable zkSNARKs and commit-and-prove extensions. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 3–33. Springer, Heidelberg, December 2021.

[CH20] Geoffroy Couteau and Dominik Hartmann. Shorter non-interactive zero-knowledge arguments and ZAPs for algebraic languages. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 768–798. Springer, Heidelberg, August 2020.

[CHM+20] Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Psi Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS.In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

[CL03] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Heidelberg, September 2003.

[CL19] Elizabeth C. Crites and Anna Lysyanskaya. Delegatable anonymous credentials from mercurial signatures. In Mitsuru Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 535–555. Springer, Heidelberg, March 2019.

[CL21] Elizabeth C. Crites and Anna Lysyanskaya. Mercurial signatures for variable-length messages. *PoPETs*, 2021(4):441–463, October 2021.

[CLP22] Aisling Connolly, Pascal Lafourcade, and Octavio Perez-Kempner. Improved constructions of anonymous credentials from structure-preserving signatures on equivalence classes. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part I*, volume 13177 of *LNCS*, pages 409–438. Springer, 2022.

[CS20] Remi Clarisse and Olivier Sanders. Group signature without random oracles from randomizable signatures. In Khoa Nguyen, Wenling Wu, Kwok-Yan Lam, and Huaxiong Wang, editors, *ProvSec 2020*, volume 12505 of *LNCS*, pages 3–23. Springer, Heidelberg, November / December 2020.

[DHS15] David Derler, Christian Hanser, and Daniel Slamanig. A new approach to efficient revocable attribute-based anonymous credentials. In Jens Groth, editor, *15th IMA International Conference on Cryptography and Coding*, volume 9496 of *LNCS*, pages 57–74. Springer, Heidelberg, December 2015.

[DS16] David Derler and Daniel Slamanig. Fully-anonymous short dynamic group signatures without encryption. Cryptology ePrint Archive, Report 2016/154, 2016.https://eprint.iacr.org/2016/154.

[DS18] David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *ASIACCS 18*, pages 551–565. ACM Press, April 2018.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar.An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

[EHK+17] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Luis Villar. An algebraic framework for Diffie-Hellman assumptions. *Journal of Cryptology*, 30(1):242–288, January 2017.

[FG18] Georg Fuchsbauer and Romain Gay. Weakly secure equivalence-class signatures from standard assumptions. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 153–183. Springer, Heidelberg, March 2018.

[FGKO17] Georg Fuchsbauer, Romain Gay, Lucas Kowalczyk, and Claudio Orlandi.Access control encryption for equality, comparison, and more. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 88–118. Springer, Heidelberg, March 2017.

[FHKS16] Georg Fuchsbauer, Christian Hanser, Chethan Kamath, and Daniel Slamanig.Practical round-optimal blind signatures in the standard model from weaker assumptions. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 391–408. Springer, Heidelberg, August / September 2016.

[FHS15] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 233–253. Springer, Heidelberg, August 2015.

[FHS19] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018.

[FP09] Georg Fuchsbauer and David Pointcheval. Proofs on encrypted values in bilinear groups and an application to anonymity of signatures. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 132–149. Springer, Heidelberg, August 2009.

[Fuc11] Georg Fuchsbauer. Commuting signatures and verifiable encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 224–245. Springer, Heidelberg, May 2011.

[Fuc14] Georg Fuchsbauer. Breaking existential unforgeability of a signature scheme from asiacrypt 2014. Cryptology ePrint Archive, Report 2014/892, 2014. https://eprint.iacr.org/2014/892.

[FV10] Georg Fuchsbauer and Damien Vergnaud. Fair blind signatures without random oracles. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 16–33. Springer, Heidelberg, May 2010.

[GHKP18] Romain Gay, Dennis Hofheinz, Lisa Kohl, and Jiaxin Pan. More efficient (almost) tightly secure structure-preserving signatures. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 230–258. Springer, Heidelberg, April / May 2018.

[GL23] Scott Griffy and Anna Lysyanskaya. Pacific: Privacy-preserving automated contact tracing scheme featuring integrity against cloning. Cryptology ePrint Archive, Paper 2023/371, 2023. https://eprint.iacr.org/2023/371.

[GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 99–108. ACM Press, June 2011.

[GWC19] Ariel Gabizon, Zachary J. Williamson, and Oana Ciobotaru. PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Paper 2019/953, 2019.

[Han23] Lucjan Hanzlik. Non-interactive blind signatures for random messages. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 722–752. Springer, Heidelberg, April 2023.

[HPP20] Chloé Hébant, Duong Hieu Phan, and David Pointcheval. Linearly-homomorphic signatures and scalable mix-nets. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 597–627. Springer, Heidelberg, May 2020.

[HRS15] Christian Hanser, Max Rabkin, and Dominique Schröder.Verifiably encrypted signatures: Security revisited and a new construction. In Günther Pernul, Peter Y. A. Ryan, and Edgar R. Weippl, editors, *ESORICS 2015, Part I*, volume 9326 of *LNCS*, pages 146–164. Springer, Heidelberg, September 2015.

[HS14] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 491–511. Springer, Heidelberg, December 2014.

[HS21] Lucjan Hanzlik and Daniel Slamanig. With a little help from my friends: Constructing practical anonymous credentials. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2004–2023. ACM Press, November 2021.

[KSD19] Mojtaba Khalili, Daniel Slamanig, and Mohammad Dakhilalian.Structure-preserving signatures on equivalence classes from standard assumptions. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 63–93. Springer, Heidelberg, December 2019.

[Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 169–189. Springer, Heidelberg, March 2012.

[LSZ22] Helger Lipmaa, Janno Siim, and Michal Zajac. Counting vampires: From univariate sumcheck to updatable ZK-SNARK. Cryptology ePrint Archive, Paper 2022/406, 2022.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *10th IMA International Conference on Cryptography and Coding*, volume 3796 of *LNCS*, pages 1–12. Springer, Heidelberg, December 2005.

[MBG+23] Omid Mir, Balthazar Bauer, Scott Griffy, Anna Lysyanskaya, and Daniel Slamanig. Aggregate signatures with versatile randomization and issuer-hiding multi-authority anonymous credentials. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 30–44. ACM Press, November 2023.

[MBKM19] Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

[MRV16] Paz Morillo, Carla Ràfols, and Jorge Luis Villar. The kernel matrix Diffie-Hellman assumption. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 729–758. Springer, Heidelberg, December 2016.

[MSBM23] Omid Mir, Daniel Slamanig, Balthazar Bauer, and René Mayrhofer.Practical delegatable anonymous credentials from equivalence class signatures. *Proc. Priv. Enhancing Technol.*, 2023(3):488–513, 2023.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk).In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, Heidelberg, August 2003.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[PM23] Colin Putman and Keith M. Martin. Selective delegation of attributes in mercurial signature credentials. In Elizabeth A. Quaglia, editor, *IMACC 2023*, volume 14421 of *LNCS*, pages 181–196. Springer, 2023.

[Poi23] David Pointcheval. Linearly-homomorphic signatures for short randomizable proofs of subset membership. In *Eighth International Joint Conference on Electronic Voting (E-Vote-ID'23)*, 2023.

[Poi24] David Pointcheval. Efficient universally-verifiable electronic voting with everlasting privacy. *Cryptology ePrint Archive*, 2024.

[RZ21] Carla Ràfols and Arantxa Zapico. An algebraic framework for universal and updatable SNARKs. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 774–804, Virtual Event, August 2021. Springer, Heidelberg.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, Heidelberg, May 1997.

[SKSW22] Yumi Sakemi, Tetsutaro Kobayashi, Tsunekazu Saito, and Riad S. Wahby.Pairing-Friendly Curves. Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-11, Internet Engineering Task Force, November 2022. Work in Progress.

[SYF+23] Rui Shi, Yang Yang, Huamin Feng, Feng Yuan, Huiqin Xie, and Jianyi Zhang. Prirpt: Practical blockchain-based privacy-preserving reporting system with rewards. *Journal of Systems Architecture*, 143:102985, 2023.

[WTSD23] Eva Wisse, Pietro Tedeschi, Savio Sciancalepore, and Roberto Di Pietro.A 2rid-anonymous direct authentication and remote identification of commercial drones. *IEEE Internet of Things Journal*, 2023.

[ZYY+23] Yonghua Zhan, Bixia Yi, Yang Yang, Rui Shi, Chen Dong, and Minming Huang.A privilege-constrained sanitizable signature scheme for e-health systems. *Journal of Systems Architecture*, 142:102939, 2023.

Dual Support Decomposition in the Head: Shorter Signatures from Rank SD and MinRank

Loïc Bidoux[1](✉), Thibauld Feneuil[2], Philippe Gaborit[3], Romaric Neveu[3], and Matthieu Rivain[2]

[1] Technology Innovation Institute, Abu Dhabi, UAE
loic.bidoux@tii.ae
[2] CryptoExperts, Paris, France
[3] University of Limoges, Limoges, France

Abstract. The MPC-in-the-Head (MPCitH) paradigm is widely used for building post-quantum signature schemes, as it provides a versatile way to design proofs of knowledge based on hard problems. Over the years, the MPCitH landscape has changed significantly, with the most recent improvements coming from *VOLE-in-the-Head* (VOLEitH) and *Threshold-Computation-in-the-Head* (TCitH).

While a straightforward application of these frameworks already improve the existing MPCitH-based signatures, we show in this work that we can adapt the arithmetic constraints representing the underlying security assumptions (here called the *modeling*) to achieve smaller sizes using these new techniques. More precisely, we explore existing modelings for the rank syndrome decoding (RSD) and MinRank problems and we introduce a new modeling, named *dual support decomposition*, which achieves better sizes with the VOLEitH and TCitH frameworks by minimizing the size of the witnesses. While this modeling is naturally more efficient than the other ones for a large set of parameters, we show that it is possible to go even further and explore new areas of parameters. With these new modeling and parameters, we obtain low-size witnesses which drastically reduces the size of the "arithmetic part" of the signature.

We apply the TCitH and VOLEitH frameworks to our new modeling for both RSD and MinRank and compare our results to the NIST candidates RYDE, MiRitH, and MIRA (MPCitH-based schemes from RSD and MinRank). We also note that recent techniques optimizing the sizes of GGM trees are applicable to our schemes and further reduce the signature sizes by a few hundred bytes. We obtain signature sizes below 3.5 kB for 128 bits of security with $N = 256$ parties (a.k.a. leaves in the GGM trees) and going as low as ≈ 2.8 kB with $N = 2048$, for both RSD and MinRank. This represents an improvement of more than 2 kB compared to the original submissions to the 2023 NIST call for additional signatures.

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15485, pp. 38–69, 2025.
https://doi.org/10.1007/978-981-96-0888-1_2

1 Introduction

The MPC-in-the-Head (MPCitH) paradigm is a popular framework to build post-quantum signatures. After sharing the secret key, the signer emulates "in his head" an MPC protocol and commits each party's view independently. He then reveals the views of a pseudo-random subset of parties, where this subset is given by the hash digest of the commitments (in the setting of the Fiat-Shamir heuristic). By the privacy of the MPC protocol, nothing is revealed about the secret key, which implies the zero-knowledge property. On the other hand, a malicious signer needs to cheat for at least one party, which shall be discovered by the verifier with high probability, hence ensuring the unforgeability property.

In the new NIST call for additional post-quantum signatures [33], many submissions rely on the MPCitH paradigm applied on a large range of security assumptions. Three MPCitH candidates fall in the rank-based cryptography category:

- RYDE [4], for which the security relies on the hardness of solving the rank syndrome decoding problem;
- MIRA [5] and MiRitH [1], for which the security relies on the hardness of solving the MinRank problem (MIRA and MiRitH rely on the same security assumption, but use different modelings and MPC protocols).

Recently, new techniques of MPC-in-the-Head have been proposed:

- the VOLE-in-the-Head (VOLEitH) framework [12] released in Summer 2023;[1]
- the TC-in-the-Head (TCitH) framework [22] released in Autumn 2023.[2]

As shown in [22] a simple application of these frameworks leads to shorter and faster signature schemes compared to those submitted to the NIST call (for similar underlying security assumption).

For MPCitH-based schemes (including those based on VOLEitH and TCitH), the signatures are composed of two parts, a "symmetric part" made of seeds and hash digests and an "arithmetic part" composed of the open party views and broadcast shares of the MPC protocol. While for a given security level the symmetric part is of rather fixed size (for the considered MPCitH framework), the arithmetic part depends on the modeling of the used security assumption and the associated MPC protocol. In the traditional broadcast-based MPCitH framework (*i.e.* the MPCitH framework widely used before VOLEitH and TCitH), to minimize the signature size, the designers had minimize the *sum of the sizes of the MPC input and of the broadcasted values* while considering only *linear* multiparty computation. With the VOLEitH and TCitH frameworks, the game rules have changed. These frameworks enable quadratic (or higher degree) multiparty

[1] While VOLEitH has not been introduced as an MPCitH technique, [22] showed that it can be considered as such.

[2] The original version of the TCitH framework was released in Autumn 2022 [23] (and published at Asiacrypt 2023), we refer here to the improved version of the TCitH framework [22].

computation, which implies that minimizing the signature size is achieved only by minimizing the MPC protocol input (i.e., the witness of the modeling).

In rank-based cryptography, several modelings for the rank syndrome decoding problem and the MinRank problem have been proposed. The first one is derived from [37] and consists in working with a permuted version and an additively-masked version of the secret. The best scheme relying on it is proposed in [15]. The second modeling is based on q-polynomials and is first used in such a context in [20]. The third modeling consists in writing the low-rank object as the product of two small matrices and is first used in such a context in [2] and [20]. The last modeling relies on Kipnis-Shamir technique, initially proposed for the cryptanalysis of the MinRank problem [29] and used to build a scheme in [1]. We sum up the different techniques to handle the rank metric in Table 1.

Table 1. Techniques used in MPCitH-based signatures for RSD and MinRank.

Problem	**Permuted Secret**	**q-Polynomial Evaluation (q-pol)**	**Matrix Rank Decomposition (MRD)**	**Kipnis Shamir (KS)**	**Dual Support Decomposition (DSD)**
RSD	BG23 [15]	RYDE [4,20]	Fen24 [20]	-	This work
MinRank	-	MIRA [5,20]	Fen24 [20]	MiRitH [1]	This work

In this work, we explore modelings for the rank syndrome decoding problem and the MinRank problems to identify the best option with the new VOLEitH and TCitH techniques. We show that the shortest signatures with RSD and MinRank are obtained thanks to the *dual support decomposition* modeling, which consists in finding a basis $(e_1, \ldots, e_r)$ and coefficients $c_{1,1}, \ldots, c_{n,r}$ such that

$$y = Hx \quad \text{and} \quad \forall i,\ x_i = \sum_{j=1}^{r} c_{i,r} \cdot e_j.$$

While this modeling is quite natural for the rank syndrome decoding problem, it requires to work with a dual matrix of a code in the MinRank problem: we need to consider the syndrome decoding problem for matrix codes. In fact, the MinRank problem being the message decoding problem for such codes, we define in this work the MinRank Syndrome problem, which is a equivalent variant of the MinRank problem which has not been previously used in cryptosystems. Working in the dual has the advantage to remove the encoded message from the witness of the code-based problem, leading to a shorter witness. With the dual support decomposition modeling, the witness size (and thus the signature size) is independent of the code dimension, thanks to the definition of the syndrome version of the MinRank problem. This enables us to optimize the parameters by taking codes of larger dimensions.

We then apply the TCitH and VOLEitH frameworks on the optimal modeling, yielding new signature schemes with smaller sizes as summarized in Table 2. We also put the signature sizes of the NIST candidates based on the same security assumptions (namely RYDE, MIRA and MiRitH) in the column "MPCitH" and their signature sizes when performing a straightforward application of VOLEitH and TCitH. We observe that the difference in signature sizes between VOLEitH and TCitH tends to disappear while increasing the parameter N, i.e., the number of leaves in GGM seed trees used for the commitment (a.k.a. the number of parties in standard MPCitH schemes). Since these two frameworks are faster than previous MPCitH schemes, it becomes natural to consider larger values of N. We obtain signature sizes down to 3.7 kB for TCitH with $N = 256$ leaves, and down to 2.9 kB for VOLEitH and TCitH with $N = 2048$ leaves (more details are given in Tables 7 and 10). The ranges of sizes reported in Table 2 correspond to a parameter N ranging between 256 and 2048. Let us note that new generic optimizations for MPCitH-based signatures have been proposed in [11] very recently. We applied these optimisations to our new signature schemes, enabling us to save an additional few hundred bytes. The obtained sizes are reported in Table 2 with the label "optimized".

Table 2. Comparison of our schemes based on dual support decomposition (DSD) with the NIST candidates based on the same security assumptions. The sizes in the column "MPCitH" are given when using seed trees with 256 leaves, while the size range in columns "VOLEitH" and "TCitH" are given when using seed trees with between 256 and 2048 leaves.

Security Assumption	Scheme	MPCitH	VOLEitH	TCitH
Rank SD	RYDE (q-pol)	5 956 B	4 133–4 720 B	4 274–5 281 B
	Our scheme (DSD), optimized	-	2 851–3 450 B	2 937–3 708 B
MinRank	MIRA (q-pol)	5 640 B	4 170–4 770 B	4 314–5 340 B
	MiRitH-Ia (KS)	5 665 B	3 762–4 226 B	3 873–4 694 B
	MiRitH-Ib (KS)	6 298 B	4 110–4 690 B	4 250–5 245 B
	Our scheme (DSD), optimized	-	2 813–3 396 B	2 896–3 640 B

Paper Organization. The paper is organized as follows: In Sect. 2, we introduce the necessary background on the rank metric and sharing schemes. We present the existing attacks against RSD and MinRank in Sect. 3. We explore the possible modelings for rank-based cryptography in Sect. 4. We recall the TCitH and VOLEitH frameworks in Sect. 5 and we apply these frameworks to the dual support decomposition modeling to obtain new signature schemes in Sect. 6.

2 Preliminaries

2.1 Notations

We denote by $\mathbb{F}_q$ the finite field of size q. The set of vectors with n coordinates in $\mathbb{F}_q$ is referred as $\mathbb{F}_q^n$, the set of matrices with m rows and n columns in

$\mathbb{F}_q$ is referred as $\mathbb{F}_q^{m\times n}$. We use lowercase bold letters to represent vectors and uppercase bold letters for matrices ($\boldsymbol{E} \in \mathbb{F}_q^{m\times n}$, $\boldsymbol{x} \in \mathbb{F}_q^k$, $x \in \mathbb{F}_q$). The subset of integers from 1 to n is represented with $[1,n]$. If S is a set, we write $x \xleftarrow{\$} S$ the uniform sampling of a random element x in S. We note the $\mathbb{F}_q$-linear subspace of $\mathbb{F}_{q^m}$ generated by $(x_1,\ldots,x_n) \in \mathbb{F}_{q^m}^n$ as $\langle x_1,\ldots,x_n\rangle$. Let us define the gaussian coefficient $\begin{bmatrix} m \\ r \end{bmatrix}_q = \prod_{i=0}^{r-1} \frac{q^m-q^i}{q^r-q^i} \approx q^{r(m-r)}$, it corresponds to the number of different dimension-r $\mathbb{F}_q$-linear subspaces of $\mathbb{F}_{q^m}$.

2.2 Secret Sharing

A threshold secret sharing scheme is a method to share a value v into a sharing $[\![v]\!] := ([\![v]\!]_1,\ldots,[\![v]\!]_N)$ such that v can be reconstructed from any $\ell+1$ shares while no information is revealed on the secret from the knowledge of ℓ shares. We note by $[\![x]\!]_i$ the i^{th} share of $[\![x]\!]$ (*i.e.* the share of the i^{th} party). We can also note $[\![x]\!]_I$ where I is a set of indices, to denote all the shares of the parties in the set I.

Let us define Shamir's secret sharing scheme [36], since the frameworks we will consider rely on it. Let ℓ and N two integers such that $1 \le \ell \le N$. Let $e, \omega_1, \ldots, \omega_N$ be $N+1$ distinct elements of $\mathbb{F} \cup \{\infty\}$. To share a value $v \in \mathbb{F}$ using Shamir's secret sharing scheme, one should

1. sample ℓ randoms values $r_1,\ldots,r_\ell$ of $\mathbb{F}$;
2. compute the polynomial P by interpolation such that

$$P(e) = v \quad \text{and} \quad \forall i \in [1,\ell], P(\omega_i) = r_i;$$

3. build the N shares $[\![v]\!]_1,\ldots,[\![v]\!]_N$ as

$$\forall i \in [1,N],\ [\![v]\!]_i := P(\omega_i).$$

To recover the secret value from $\ell+1$ shares, we re-compute the polynomial P by interpolation and we just deduce $P(e)$. Let us stress that $P(\infty)$ refers to the leading coefficient of the polynomial P. The most classical choice is to set e to zero but we may consider alternative choices depending on the context (and in particular $e=\infty$).

We define the *degree* of a Shamir's secret sharing as the degree of the underlying polynomial. A sharing generated using the above process is of degree ℓ. The sum of a d_1-degree sharing and a d_2-degree sharing is of degree $\max(d_1,d_2)$, while the multiplication is of degree d_1+d_2.

2.3 Rank Metric and Hard Problems for Cryptography

We will first recall some background on the Rank Metric, and we will then define hard problems we will use (RSD and $\mathsf{MinRank}$).

Definition 1 (Rank Metric over $\mathbb{F}_{q^m}^n$). *Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$, and $\mathcal{B} = (b_1, \ldots, b_m) \in \mathbb{F}_{q^m}^m$ an $\mathbb{F}_q$-basis of $\mathbb{F}_{q^m}$. Each coordinate x_j can be associated with a vector $(x_{j,1}, \ldots, x_{j,m}) \in \mathbb{F}_q^m$ such that $x_j = \sum_{i=1}^m x_{j,i} b_i$. Let us define the following notations:*

- $\boldsymbol{M_x} = (x_{j,i})_{(j,i) \in [1,n] \times [1,m]}$ *is the* matrix associated to the vector $\boldsymbol{x}$*;*
- *the* rank weight *is defined as:* $w_R(\boldsymbol{x}) = \mathsf{rank}(\boldsymbol{M_x})$*;*
- *the* distance *between two vectors* $\boldsymbol{x}$ *and* $\boldsymbol{y}$ *in* $\mathbb{F}_{q^m}^n$ *is:* $d(x, y) = w_R(\boldsymbol{x} - \boldsymbol{y})$*;*
- *the* support *of a vector* $\mathrm{Supp}(\boldsymbol{x})$ *is the* $\mathbb{F}_q$*-linear subspace of* $\mathbb{F}_{q^m}$ *generated by its coordinates:* $\mathrm{Supp}(\boldsymbol{x}) = \langle x_1, \ldots, x_n \rangle$.

Definition 2. *A linear code $\mathcal{C}$ over $\mathbb{F}_{q^m}$ of dimension k and length n is a linear subspace of $\mathbb{F}_{q^m}^n$ of dimension k. The elements of $\mathcal{C}$ are called codewords. The code $\mathcal{C}$ can be represented in two ways:*

- *by a generator matrix $\boldsymbol{G}$, where $\mathcal{C} = \{\boldsymbol{mG}, \boldsymbol{m} \in \mathbb{F}_{q^m}^k\}$, or*
- *by a parity-check matrix $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ where $\mathcal{C} = \{\boldsymbol{x} \in \mathbb{F}_{q^m}^n : \boldsymbol{H}\boldsymbol{x}^\top = \boldsymbol{0}^\top\}$*

We now continue by formally recalling the definition of the rank syndrome decoding (RSD) problem.

Definition 3 (RSD problem). *Let q, m, n, k and r be positive integers. Let $\boldsymbol{H} \xleftarrow{\$} \mathbb{F}_{q^m}^{(n-k)\times n}$ and $\boldsymbol{x} \xleftarrow{\$} \mathbb{F}_{q^m}^n$ such that $w_R(\boldsymbol{x}) = r$. Let $\boldsymbol{y}^\top = \boldsymbol{H}\boldsymbol{x}^\top$. Given $(\boldsymbol{H}, \boldsymbol{y})$, the computational $\mathsf{RSD}(q, m, n, k, r)$ problem asks to find a vector $\tilde{\boldsymbol{x}} \in \mathbb{F}_{q^m}^n$ such that $\boldsymbol{H}\tilde{\boldsymbol{x}}^\top = \boldsymbol{y}^\top$ and $w_R(\tilde{\boldsymbol{x}}) = r$.*

We now introduce a variant of the above problem, the RSD_s problem and later argue that it is as hard as the standard RSD problem.

Definition 4 (RSD_s problem). *Let q, m, n, k and r be positive integers. Let $\boldsymbol{H} \xleftarrow{\$} \mathbb{F}_{q^m}^{(n-k)\times n}$ and $\boldsymbol{x} = (x_i) \xleftarrow{\$} \mathbb{F}_{q^m}^n$ such that $w_R(\boldsymbol{x}) = r$, $x_1 = 1 \in \mathbb{F}_{q^m}$ and $\langle x_1, \ldots, x_r \rangle_{\mathbb{F}_q} = \mathrm{Supp}(\boldsymbol{x})$. Let $\boldsymbol{y}^\top = \boldsymbol{H}\boldsymbol{x}^\top$. Given $(\boldsymbol{H}, \boldsymbol{y})$, the computational $\mathsf{RSD}_s(q, m, n, k, r)$ problem asks to find a vector $\tilde{\boldsymbol{x}} \in \mathbb{F}_{q^m}^n$ such that $\boldsymbol{H}\tilde{\boldsymbol{x}}^\top = \boldsymbol{y}^\top$ and $w_R(\tilde{\boldsymbol{x}}) = r$.*

The last problem we will rely on is the well-known $\mathsf{MinRank}$ problem:

Definition 5 ($\mathsf{MinRank}$ problem). *Let q, m, n, k and r be positive integers. Let $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_k, \boldsymbol{E} \in \mathbb{F}_q^{m\times n}$ and $\boldsymbol{x} := (x_1, \ldots, x_k) \in \mathbb{F}_q^k$ be uniformly sampled such that*

$$\mathsf{rank}(\boldsymbol{E}) \leq r \quad \textit{with} \quad \boldsymbol{M} := \boldsymbol{E} - \sum_{i=1}^{k} x_i \boldsymbol{M}_i.$$

Given $\boldsymbol{M}, \boldsymbol{M}_1 \ldots, \boldsymbol{M}_k$, the computational $\mathsf{MinRank}(q, m, n, k, r)$ problem asks to retrieve the vector $\boldsymbol{x}$.

The last notion to recall is the Gilbert-Varshamov bound for the rank metric and for MinRank. This bound in rank metric has been introduced in [30]. It can be seen as the probable minimum weight of a random code.

Definition 6 (Rank Gilbert-Varshamov Bound). *Let S_r be the number of elements of the sphere in $\mathbb{F}_{q^m}^n$ of radius r centered in 0, i.e., the number of elements in $\mathbb{F}_{q^m}^n$ of weight exactly r. We have $S_0 = 1$, and for $r \geq 1$,*

$$S_r = \prod_{j=0}^{r-1} \frac{(q^n - q^j)(q^m - q^j)}{q^r - q^j}.$$

Let $B_r := \sum_{i=0}^{r} S_r$ be the number of elements of the ball in $\mathbb{F}_{q^m}^n$ of radius r centered in 0. The Rank Gilbert-Varshamov (RGV) bound for an $[n, k]$ linear code over $\mathbb{F}_{q^m}$ is the smallest integer r such that

$$q^{m(n-k)} \leq B_r$$

Using the approximation $B_r \approx q^{(m+n-r)r}$, one can say the RGV bound is the smallest r such that $m(n-k) \leq (m-r)r + nr$. We call this value d_{RGV}. The same bound exists for matrix codes (i.e., for MinRank) as they are simply $\mathbb{F}_q$-linear codes. Courtois described this bound in [17, Section 24.2], and it can also be derived from the one above easily (consider a $[m \times n, k]$ linear code over $\mathbb{F}_q$ instead of $[n, k]$ linear over $\mathbb{F}_{q^m}$). This bound is also mentioned in attacks on MinRank ([8,9] for instance). Concretely, this states that, for an instance of MinRank with parameters (q, m, n, k, r), we do not expect to obtain more than one solution if r is chosen such that $k + 1 \leq (m - r)(n - r)$.

Complexity of Attacks for Parameters on the GV Bound. For RSD, the parameter r is taken as $d_{\mathsf{RGV}} - 1$, i.e., the highest r such that $(m-r)r + nr < m(n-k)$. With this parameter, if $\boldsymbol{H}$ and $\boldsymbol{y}$ were to be randomly sampled, one would expect to have a solution with probability $q^{(m+n-r)r-m(n-k)}$. Since $\boldsymbol{y}$ is set so there is a solution and since we are below RGV, it is not expected to have another solution. For MinRank, we take parameters on the RGV bound, with $k+1 = (m-r)(n-r)$. For $k+1$ matrices randomly sampled $(\boldsymbol{M}, \boldsymbol{M}_1, \ldots, \boldsymbol{M}_k)$, the probability to have a solution to the MinRank instance is $q^{(m+n-r)r-(mn-k)}$. Since $\boldsymbol{M}$ is set so that there is a solution and since we are on GV, it is not expected to have another solution for the instance. Let us now explain why in addition to having only one solution, it is important to take parameters according to these bounds. Since the combinatorial attacks from [34] for RSD and [26] for MinRank, very few improvements have been made in the complexity. For MinRank, the kernel attack is still the best combinatorial attack, and for RSD, the exponential part of the complexities is still quadratic and has known almost no improvement for over 20 years (with the exception of [6], which slightly improved the complexity). Regarding the algebraic attacks, introduced in [7] and improved in [10] and [8], they managed to greatly reduce the complexity for the RQC and LRPC schemes. However, this came from the fact that these parameters were not on RGV. The attacked parameters were in $\mathcal{O}\left(\sqrt{n-k}\right)$, which made them easier to attack, whereas we will considerparameters around the RGV bound, in $\mathcal{O}(n)$. In practice, for parameters taken at the RGV bound, or just below, the algebraic attacks have roughly the same complexity as the combinatorial ones ([8]). Overall, this means that, for parameters taken on the Rank Gilbert-Varshamov bound, the attacks have known no significant amelioration for the last 20 years.

3 Security and Parameters for RSD_s and MinRank

We give here the well known reduction from RSD to RSD_s, and then the attacks considered against RSD and MinRank, which we will use in order to establish parameters for the signature schemes. We will also use these attacks in order to establish parameters to compare the different modelings in Sect. 4.

3.1 Security of the Rank Syndrome Decoding Problem

We deal here with the RSD problem, first by explaining the relation between RSD and RSD_s, and then the attacks on RSD.

Security Reduction. The RSD_s problem was most notably used in the RQC scheme in order to optimize it [31]. In the following, we show that the RSD_s problem is as hard as the standard RSD problem. More precisely, we show that any RSD instance can be solved by an RSD_s solver. This is the same reduction as in [6,7,34], and others, used to specialize some variables. We exhibit below the reduction which has not formally been described in previous works (as part of the folklore of rank-based cryptography).

Proposition 1. *Let q, m, n, k, r be positive integers such that $n > k$. Let $\mathcal{A}_s$ be an algorithm which solves a $(q, m, n, k+1, r)$-instance of the RSD_s problem in time t with success probability ε_s. Then there exists an algorithm $\mathcal{A}$ which solves a (q, m, n, k, r)-instance of the* RSD *problem in time t with probability ε, where*

$$\varepsilon \geq \left(\prod_{i=0}^{r-1} \frac{q^n - q^{n-r+i}}{q^n - q^i}\right) \cdot \varepsilon_s$$

under the assumption that the code $\mathcal{C}$ associated to the parity-check matrix $\boldsymbol{H}$ of the RSD *instance contains no words of weight r.*

Proof. See the full version [16].

Remark 1. In practice, the loss factor in Proposition 1 tends to 1 when q grows. For our considered parameters, with $q = 2$, its value is around 0.3. Moreover, one can get the average number of codewords of $\mathcal{C}$ of weight r to justify our assumption. Let $\mathcal{S}_r = \prod_{i=0}^{r-1} \frac{(q^n-q^i)(q^m-q^i)}{q^r-q^i}$ be the number of words in $\mathbb{F}_{q^m}^n$ of weight exactly r. Then, on average, there are $\frac{S_r}{q^{m(n-k)}}$ words of rank r in the code. When below RGV, this makes the probability that a random code $\mathcal{C}$ contains no codeword of weight r close to 1.

Remark 2. The best known attacks on RSD use the reduction to RSD_s in order to solve the instance [6,8,10,34], meaning that in practice we consider the best attacks on RSD to evaluate the security of RSD_s.

3.2 Parameters Choice for $\mathsf{RSD_s}$

Because of the space constraints, we recall the best attacks on RSD in the full version [16]. According to these attacks, we give in Table 3 the parameters which we will use for our $\mathsf{RSD_s}$ instances.

Table 3. Choice of parameters for $\mathsf{RSD_s}$

NIST Security level	q	m	n	k	r
I	2	53	53	45	4
III	2	79	75	67	4
V	2	97	95	87	4

3.3 Parameters Choice for MinRank

Because of the space constraints, we recall the attacks on MinRank in the full version [16]. According to these attacks, we give in Table 4 the parameters which we will use for our MinRank instances.

Table 4. Choice of parameters for MinRank

NIST Security level	q	m	n	k	r
I	2	43	43	1520	4
III	2	60	60	3135	4
V	2	75	75	5040	4

4 MPCitH Modeling for $\mathsf{RSD_s}$ and MinRank

A zero-knowledge proof constructed using the MPCitH paradigm is composed of two parts, a "symmetric part" made of GGM trees (or Merkle trees) and an "arithmetic part" composed of the open party views and broadcast shares of the MPC protocol. While for a given security level the symmetric part is of rather fixed size (e.g., around 2kB for GGM trees and 4kB for Merkle trees at a 128-bit security level), the arithmetic part depends on the modeling (i.e., the way the problem instance is verified) and the associated MPC protocol. For the recent TCitH and VOLEitH techniques, the arithmetic part is actually mainly impacted by the size of the witness, which favors modelings with low-size witnesses.

In this section, we study different modelings for RSD and MinRank with respect to the witness size criterion. For the RSD problem, we recall the permuted secret, q-polynomial and Kipnis-Shamir modelings. We propose an other modeling, named *dual support decomposition*, which can be seen as an improvement of the rank decomposition from [20]. We also slightly improve all the modelings by relying on the $\mathsf{RSD_s}$ variant. For the MinRank problem, we recall the q-polynomial and Kipnis-Shamir modelings and propose an adaptation of the dual support decomposition modeling for MinRank.

4.1 Modelings for the $\mathsf{RSD_s}$ Problem

Permuted Secret. We start by recalling the permuted secret technique, which was used for RSD in [15]. The idea of this technique consists in revealing a "permuted" and a "masked" versions of the secret: let us denote σ an isometry in the rank metric (such a isometry consists of multiplying the secret matrix by a invertible matrices on both sides) and $\boldsymbol{u}$ a vector of the left kernel of $\boldsymbol{H}$, one reveals $\boldsymbol{v} := \sigma(\boldsymbol{x})$ and $\tilde{\boldsymbol{x}} := \boldsymbol{x} + \boldsymbol{u}$ and the goal is to find such values σ and $\boldsymbol{u}$. More precisely, the rank syndrome decoding problem consists, from two vectors $\boldsymbol{v}, \tilde{\boldsymbol{x}} \in \mathbb{F}_{q^m}^n$ satisfying $w_R(\boldsymbol{v}) = r$ and $\boldsymbol{H}\tilde{\boldsymbol{x}}^\top = \boldsymbol{y}^\top$, in finding an isometry σ and a vector $\boldsymbol{u} \in \mathbb{F}_{q^m}^n$ such that

$$\begin{cases} \boldsymbol{H}\boldsymbol{u}^\top = \boldsymbol{0}^\top, \\ \sigma(\tilde{\boldsymbol{x}}) = \boldsymbol{v} + \sigma(\boldsymbol{u}). \end{cases}$$

Indeed, if we get both σ and $\boldsymbol{u}$, we can easily restore the initial secret as $\boldsymbol{x} := \tilde{\boldsymbol{x}} - \boldsymbol{u}$: we have $\boldsymbol{H}\boldsymbol{x}^\top = \boldsymbol{y}^\top - \boldsymbol{0}^\top$ and $w_R(\boldsymbol{x}) = w_R(\sigma(\boldsymbol{x})) = w_R(\sigma(\tilde{\boldsymbol{x}}) - \sigma(\boldsymbol{u})) = w_R(\boldsymbol{v}) = r$.

Unfortunately, this modeling is not compatible with the recent MPCitH techniques as TCitH or VOLEitH. Such techniques requires at least additive sharings over a commutative group (or for the more recent techniques, Shamir's secret sharing over a ring). However, the isometry σ lives in a non-commutative group, so it requires to rely on a special form of MPCitH named the shared-permutation framework [15,21].

Rank Decomposition. The Rank Decomposition protocol proposed in [20] aims to verify the rank of the witness by using the rank decomposition theorem. In this modeling, the rank syndrome decoding problem consists in finding a vector $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ and two matrices $\boldsymbol{T} \in \mathbb{F}_q^{m \times r}$ and $\boldsymbol{R} \in \mathbb{F}_q^{r \times n}$ such that

$$\boldsymbol{H}\boldsymbol{x}^\top = \boldsymbol{y}^\top \quad \text{and} \quad \boldsymbol{X} = \boldsymbol{T}\boldsymbol{R},$$

where $\boldsymbol{X} \in \mathbb{F}_q^{m \times n}$ is the matrix associated to $\boldsymbol{x}$. Concretely, the MPC protocol takes as input some shares of $\boldsymbol{x}$, of $\boldsymbol{T}$ and of $\boldsymbol{R}$. The protocol then checks that $\boldsymbol{X} = \boldsymbol{T}\boldsymbol{R}$, where the shares of $\boldsymbol{X}$ is derived from those of $\boldsymbol{x}$. Using the standard representation $\boldsymbol{H} = \left(\boldsymbol{I_{n-k}} || \boldsymbol{H'}\right)$, one can send only the right part of $\boldsymbol{x}$ of size k, denoted as $\boldsymbol{x_B}$. Furthermore, it is possible to send one less column of the matrix $\boldsymbol{T}$, since $1 \in \text{Supp}\,(\boldsymbol{x})$ (see [4] for the optimization) and as a result the size of witness is (in bits):

$$(\underbrace{k \cdot m}_{\boldsymbol{x_B}} + \underbrace{(r-1) \cdot m}_{\boldsymbol{T}} + \underbrace{r \cdot (n-r)}_{\boldsymbol{R}}) \cdot \log_2(q).$$

q-Polynomial. The q-polynomial technique proposed in [20] to check the rank metric constitutes an improvement compared to a number of previous methods. Let us first recall the definition of a q-polynomial.

Definition 7 (q-polynomial). *A q-polynomial of q-degree r is a polynomial in $\mathbb{F}_{q^m}[X]$ of the form:*

$$P(X) = X^{q^r} + \sum_{i=0}^{r-1} p_i \cdot X^{q^i} \qquad \textit{with } p_i \in \mathbb{F}_{q^m}.$$

The roots of a q-polynomial of q-degree r form a linear subspace of $\mathbb{F}_{q^m}$ of dimension at most r. Moreover, for each linear subspace of $\mathbb{F}_{q^m}$ of dimension at most r, there exists a unique monic q-polynomial of q-degree r annihilating all the elements of the subspace. Let $\boldsymbol{x} = (x_1, \dots, x_n) \in \mathbb{F}_{q^m}^n$ of rank $w_R(\boldsymbol{x}) = r$ and let $P_{\boldsymbol{x}}(X)$ the monic q-polynomial annihilating Supp $(\boldsymbol{x})$. In this modeling, the rank syndrome decoding problem consists in finding a vector $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ and a q-polynomial $P_{\boldsymbol{x}} \in \mathbb{F}_{q^m}[X]$ of q-degree r such that

$$\boldsymbol{H}\boldsymbol{x}^\top = \boldsymbol{y}^\top \quad \text{and} \quad \forall i\ , P_{\boldsymbol{x}}(x_i) = 0.$$

Concretely, the MPC protocol based on the q-polynomial technique takes as input some shares of $\boldsymbol{x}$ and some shares of $P_{\boldsymbol{x}}(X)$. The protocol then checks that $P_{\boldsymbol{x}}(x_i) = 0$ for all $i \in [1, n]$. As previously, one can send only the right part of $\boldsymbol{x}$ of size k using the standard representation of $\boldsymbol{H}$. Furthermore, it is possible to send one less coefficient of the polynomial $P_{\boldsymbol{x}}$ when relying on RSD_s. As a result the size of witness is (in bits):

$$(\underbrace{k \cdot m}_{\boldsymbol{x_B}} + \underbrace{(r-1) \cdot m}_{P_{\boldsymbol{x}}}) \cdot \log_2(q).$$

This modeling based on q-polynomials currently leads to the shortest communications for RSD when considering *linear* multiparty computation, but it is not the best one when considering non-linear multiparty computation as in the new MPCitH frameworks.

Kipnis-Shamir. Historically, the Kipnis-Shamir modeling was introduced in the cryptanalysis of the $\mathsf{MinRank}$ problem [29]. We can use the same idea to have a modeling of RSD. It consists in giving the right-kernel of the matrix of $\boldsymbol{x}$. We denote this matrix in $\mathbb{F}_q^{m \times n}$ by $\boldsymbol{X}$. If $w_R(\boldsymbol{x}) = r$, then the right-kernel of $\boldsymbol{X}$ is of dimension $n - r$ and can be represented by an $r \times (n-r)$ matrix.

In the RSD_s case, the witness is composed of $\boldsymbol{x}$ and of the matrix $\boldsymbol{A} \in \mathbb{F}_q^{r \times (n-r)}$. The MPC protocol takes as input $\boldsymbol{K} = \begin{pmatrix} \boldsymbol{I_{n-r}} \\ \boldsymbol{A} \end{pmatrix}$, and then checks that $\boldsymbol{X}\boldsymbol{K} = \boldsymbol{0}$. It is possible to send only $\boldsymbol{x_B}$, as previously with q-polynomials, and since 1 is in the support, the size of the witness is:

$$(\underbrace{k \cdot m}_{\boldsymbol{x_B}} + \underbrace{(r-1) \cdot (n-r)}_{\boldsymbol{A}}) \cdot \log_2(q) \ .$$

Note that transmitting $\boldsymbol{A}$ costs $(r-1)\cdot(n-r)$ only since we know that 1 is in $\boldsymbol{x}$. This approach is slightly better than the q-polynomial technique in terms of witness size.

Dual Support Decomposition. Finally, we introduce an other modeling for $\mathsf{RSD_s}$, using only the support and the coordinates. This can be seen as an improvement of the rank decomposition from [20], as our new modeling does not rely on having $\boldsymbol{x_B} \in \mathbb{F}_{q^m}^k$ as input. To that end, one has as inputs:

– The support of $\boldsymbol{x}$, $\text{Supp}(\boldsymbol{x}) = \langle 1, x_2, \ldots, x_r \rangle$;
– The coordinates of $\boldsymbol{x}$ in this basis, i.e., $\boldsymbol{C} \in \mathbb{F}_q^{r\times(n-r)}$ such that

$$(1, x_2, \ldots, x_r)\cdot\left(\boldsymbol{I_r}\ \boldsymbol{C}\right) = (1, x_2, \ldots, x_n) = \boldsymbol{x}$$

More precisely, in this modeling, the $\mathsf{RSD_s}$ problem consists in finding $x_2, \ldots, x_r \in \mathbb{F}_{q^m}$ and $\boldsymbol{C} \in \mathbb{F}_q^{r\times(n-r)}$ such that

$$\boldsymbol{H}\boldsymbol{x}^\top = \boldsymbol{y}^\top \quad \text{where} \quad \boldsymbol{x} := (1, x_2, \ldots, x_r)\cdot\left(\boldsymbol{I_r}\ \boldsymbol{C}\right).$$

Concretely, after computing $\boldsymbol{x} = (x_1, \ldots, x_r)\cdot\left(\boldsymbol{I_r}\ \boldsymbol{C}\right)$, one verifies that $\boldsymbol{H}\boldsymbol{x}^T$ is indeed equal to $\boldsymbol{y}^T$. Since 1 is in the support of $\boldsymbol{x}$, it is possible to transmit only $r-1$ elements for $\text{Supp}(\boldsymbol{x})$, and we can have a gain on the matrix $\boldsymbol{C}$ as well since the r first coordinates are linearly independent. This results in an efficient protocol, where the inputs are of size

$$(\underbrace{(r-1)\cdot m}_{\text{Supp}(\boldsymbol{x})} + \underbrace{r\cdot(n-r)}_{\boldsymbol{C}})\cdot\log_2(q)$$

We see here that the input size does not depend on k anymore, allowing us to take more efficient parameters.

Global Comparison. Table 5 provides a global comparison of the different modelings in terms of witness size for the RSD problem. For each of the described modelings, we provide the size formula as well as the obtained concrete size for optimized parameters reaching a 128-bit security according to the attacks in Sect. 3.2.

Table 5. Witness size for different MPCitH modelings for the $\mathsf{RSD_s}$ problem.

Modeling	**Witness size**	**Parameters for $\lambda = 128$**	
		(q, m, n, k, r)	**Size**
Rank Decomposition	$[\,km + (r-1)m + r(n-r)\,]\cdot\log_2(q)$	$(2, 31, 33, 15, 10)$	122 B
q-polynomial	$[\,km + (r-1)m\,]\cdot\log_2(q)$	$(2, 31, 33, 15, 10)$	93 B
Kipnis-Shamir	$[\,km + (r-1)(n-r)\,]\cdot\log_2(q)$	$(2, 31, 33, 15, 10)$	86 B
Dual Support Decomp.	$[\,(r-1)m + r(n-r)\,]\cdot\log_2(q)$	$(2, 53, 53, 45, 4)$	45 B

4.2 Modelings for the MinRank Problem

The MinRank problem is closely related to the RSD problem. The two problems indeed share a number of similarities as evidence of the algebraic attacks applying to both problems (see, e.g., [8,10]). Quite naturally, most of the above modelings for RSD can be adapted for MinRank.

Rank Decomposition. Similar to RSD_s, the Rank Decomposition protocol was introduced in [20]. This protocol takes as input (shares of) $\boldsymbol{x} \in \mathbb{F}_q^k$, $\boldsymbol{T} \in \mathbb{F}_q^{m\times r}$, and $\boldsymbol{R} \in \mathbb{F}_q^{r\times n}$. Then, after building $\boldsymbol{E} = \boldsymbol{M} + \sum_{i=1}^k x_i \boldsymbol{M}_i$, the protocol checks that $\boldsymbol{E} = \boldsymbol{TR}$. This leads to a witness size (in bits) of

$$(\underbrace{k}_{\boldsymbol{x}} + \underbrace{r\cdot(m-r)}_{\boldsymbol{T}} + \underbrace{r\cdot n}_{\boldsymbol{R}})\cdot \log_2(q).$$

q-Polynomial. The q-polynomial technique of [20] can be also applied to MinRank: the witness is composed of the shares of $\boldsymbol{x} \in \mathbb{F}_q^k$ and the coefficients $\boldsymbol{\beta} \in \mathbb{F}_{q^m}^r$ of the q-polynomial associated to $\boldsymbol{E}$. The MPC protocol computes $\boldsymbol{E} = \boldsymbol{M} + \sum_{i=1}^k x_i \boldsymbol{M}_i$ and verifies that $P_{\boldsymbol{E}}(X) := \sum_{i=0}^{r-1} \beta_i X^{q^i} + X^{q^r}$ is the annihilator polynomial of $\boldsymbol{E}$. This verification relies on the isomorphism between $\mathbb{F}_{q^m}$ and $\mathbb{F}_q^m$, and associates each column of $\boldsymbol{E}$, denoted as $\boldsymbol{e}_i$, to an element of $\mathbb{F}_{q^m}$, e_i. The protocol hence simply checks that $P_{\boldsymbol{E}}(e_i) = 0$ for $i \in [1, n]$.

With this modeling, the size of the witness size is (in bits):

$$(\underbrace{k}_{\boldsymbol{x}} + \underbrace{r\cdot m}_{P_{\boldsymbol{E}}})\cdot \log_2(q)\ .$$

Kipnis-Shamir. This is the modeling used in MiRitH [1], which is an improvement of MinRank-in-the-Head [2]. The goal of this modeling is to use the right kernel of $\boldsymbol{E}$ in order to prove its rank. Let $\boldsymbol{K} = \begin{bmatrix} \boldsymbol{I}_{n-r} \\ \boldsymbol{A} \end{bmatrix}$ a matrix of rank $n-r$ representing the right kernel of $\boldsymbol{E}$. The witness is composed of $\boldsymbol{x}$ and $\boldsymbol{A} \in \mathbb{F}_q^{r\times(n-r)}$. The protocol recomputes $\boldsymbol{E} = \boldsymbol{M} + \sum_{i=1}^k x_i \boldsymbol{M}_i$ and verifies that $\boldsymbol{E}\cdot\boldsymbol{K} = \boldsymbol{0}$. If the verification succeeds, one deduces that $\boldsymbol{E}$ is indeed of rank r since it has a kernel of rank $n-r$.

With this modeling, the witness is of size:

$$(\underbrace{k}_{\boldsymbol{x}} + \underbrace{r\cdot(n-r)}_{\boldsymbol{A}})\cdot \log_2(q)\ .$$

As for RSD_s, the witness is smaller with this modeling than with the q-polynomials technique.

New Modeling for the MinRank *Problem: Dual Support Decomposition.* We introduce hereafter a new MPCitH modeling for the MinRank problem which achieves smaller witness sizes than the previous modelings. To build our modeling, we will rely on an alternative formulation of this problem, namely, its syndrome version, which has not been previously used to build cryptosystems. More precisely, this problem can be expressed in a syndrome decoding way by using the dual of the matrix code generated by $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_k$. First, one can define the map

$$\begin{array}{cccc} \rho : & \mathbb{F}_q^{m \times n} & \to & \mathbb{F}_q^{mn} \\ & \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & & \vdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix} & \mapsto & \left(a_{1,1}, \ldots, a_{1,n}, \ldots, a_{m,1}, \ldots, a_{m,n}\right) . \end{array}$$

The generator matrix of the code $\langle \boldsymbol{M}_1, \ldots, \boldsymbol{M}_k \rangle$ is the matrix $\boldsymbol{G} \in \mathbb{F}_q^{k \times mn}$ whose lines are $\rho(\boldsymbol{M}_1), \ldots, \rho(\boldsymbol{M}_k)$, i.e.,

$$\boldsymbol{G} = \begin{pmatrix} \rho(\boldsymbol{M}_1) \\ \vdots \\ \rho(\boldsymbol{M}_k) \end{pmatrix} .$$

Solving the MinRank problem then consists, given $\boldsymbol{G}$ and $\boldsymbol{M}$, in finding $\boldsymbol{x}$ and $\boldsymbol{E}$ such that $\rho(\boldsymbol{M}) = -\boldsymbol{x}\boldsymbol{G} + \rho(\boldsymbol{E})$ and $\mathsf{rank}(\boldsymbol{E}) \leq r$. The dual matrix of this code is, as usual, a full-rank matrix $\boldsymbol{H} \in \mathbb{F}_q^{(mn-k) \times mn}$ such that $\boldsymbol{G}\boldsymbol{H}^\top = \boldsymbol{0}$. Then, the MinRank Syndrome problem can be defined by applying $\boldsymbol{H}^\top$ in the linear constraint: given $\boldsymbol{H}$ and $\boldsymbol{y} := \rho(\boldsymbol{M})\boldsymbol{H}^\top$, the MinRank Syndrome problem consists in finding $\boldsymbol{E}$ such that $\boldsymbol{y} = \boldsymbol{0} + \rho(\boldsymbol{E})\boldsymbol{H}^\top$ and $\mathsf{rank}(\boldsymbol{E}) \leq r$.

Definition 8 (MinRank Syndrome **problem).** *Let q, m, n, k and r be positive integers. Let $\boldsymbol{H} := \left[\boldsymbol{I}_{mn-k}\ \boldsymbol{H'}\right] \in \mathbb{F}_q^{(mn-k) \times mn}$ where $\boldsymbol{H'} \in \mathbb{F}_q^{(mn-k) \times k}$, and $\boldsymbol{y} \in \mathbb{F}_q^{mn-k}$. The* MinRank Syndrome *problem asks to find $\boldsymbol{E}$ such that $\rho(\boldsymbol{E})\boldsymbol{H}^\top = \boldsymbol{y}$ and* $\mathsf{rank}(\boldsymbol{E}) \leq r$.

Proposition 2. *The* MinRank *problem and the* MinRank Syndrome *problem are equivalent.*

Proof. The proof is similar to the proof of equivalence between the decoding problem and the syndrome decoding problem for the Hamming metric.

- Let us explain how to solve a MinRank Syndrome instance $(\boldsymbol{H}, y)$ using a MinRank solver. The first step consists in finding a solution $\rho(\boldsymbol{M})$ to the linear system $\rho(\boldsymbol{M})\boldsymbol{H}^\top = \boldsymbol{y}$ without any constraint on the rank of $\boldsymbol{M}$. Then, we consider a dual matrix $\boldsymbol{G}$ of $\boldsymbol{H}$, where we interpret each row as a flatten matrix among $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_k$. We can then find $\boldsymbol{x}$ and $\boldsymbol{E}$ such that $\boldsymbol{M} + \sum_{i=1}^k x_i \boldsymbol{M}_i = \boldsymbol{E}$ and $\mathsf{rank}(\boldsymbol{E}) \leq r$ using the MinRank solver. Such an $\boldsymbol{E}$ is directly a solution of our MinRank Syndrome instance, since

$$\rho(\boldsymbol{E})\boldsymbol{H}^\top = [-\boldsymbol{x}\boldsymbol{G} + \rho(\boldsymbol{E})]\boldsymbol{H}^\top = \rho(\boldsymbol{M})\boldsymbol{H}^\top = \boldsymbol{y}.$$

– Let us now explain how to solve a MinRank instance $(\boldsymbol{M}, \boldsymbol{M}_1, \ldots, \boldsymbol{M}_k)$ using a MinRank Syndrome solver. First, we can build the dual matrix $\boldsymbol{H}$ from $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_k$, such that $\boldsymbol{G}\boldsymbol{H}^\top = \boldsymbol{0}$ where the rows of $\boldsymbol{G}$ are defined as $\rho(\boldsymbol{M}_1), \ldots, \rho(\boldsymbol{M}_k)$. We can then find $\boldsymbol{E}$ such that $\rho(\boldsymbol{E})\boldsymbol{H}^\top = \rho(\boldsymbol{M})\boldsymbol{H}^\top$ and $\mathsf{rank}(\boldsymbol{E}) \leq r$ using the MinRank Syndrome solver (defining the syndrome as $\boldsymbol{y} = \rho(\boldsymbol{M})\boldsymbol{H}^\top$). Therefore, $\rho(\boldsymbol{E} - \boldsymbol{M})$ is in the right kernel of $\boldsymbol{H}$, for which a basis is described by the rows of $\boldsymbol{G}$. By Gaussian elimination, we can thus find $\boldsymbol{x}$ such that $\rho(\boldsymbol{E} - \boldsymbol{M}) = \boldsymbol{x}\boldsymbol{G}$, i.e., such that $\rho(\boldsymbol{E} - \boldsymbol{M}) = \sum_{i=1}^{k} x_i \cdot \rho(\boldsymbol{M}_i)$. Such an $\boldsymbol{x}$, together with $\boldsymbol{E}$, forms a solution of our MinRank instance.

□

In the *Dual Support Decomposition* modeling, we rely on the MinRank Syndrome problem instead of the standard MinRank problem. In this modeling, the protocol aims at verifying that a matrix $\boldsymbol{E}$ is solution of the constraints $\rho(\boldsymbol{E})\boldsymbol{H}^\top = \boldsymbol{y}$ and $\mathsf{rank}(\boldsymbol{E}) \leq r$ for some $\boldsymbol{H} \in \mathbb{F}_q^{(mn-k)\times mn}$ and $\boldsymbol{y} \in \mathbb{F}_q^{mn-k}$. As in the rank decomposition method from [20], one can view $\boldsymbol{E}$ as a product of two matrices, $\boldsymbol{E} = \boldsymbol{S}\boldsymbol{C}$, with $\boldsymbol{S} \in \mathbb{F}_q^{m\times r}$ and $\boldsymbol{C} \in \mathbb{F}_q^{r\times n}$. Furthermore, one can write without loss of generality $\boldsymbol{S}$ as $\begin{bmatrix} \boldsymbol{I}_r \\ \boldsymbol{S}' \end{bmatrix}$ for some matrix $\boldsymbol{S}' \in \mathbb{F}_q^{(m-r)\times r}$ (this is always possible up to a permutation of the lines). Then, one can simply set $\boldsymbol{E} = \begin{bmatrix} \boldsymbol{I}_r \\ \boldsymbol{S}' \end{bmatrix} \cdot \boldsymbol{C}$. Therefore, given $\boldsymbol{S}'$ and $\boldsymbol{C}$, the protocol simply checks:

$$\rho(\boldsymbol{S}\boldsymbol{C})\boldsymbol{H}^\top = \boldsymbol{y} \quad \text{with} \quad \boldsymbol{S} = \begin{bmatrix} \boldsymbol{I}_r \\ \boldsymbol{S}' \end{bmatrix}.$$

The use of the dual form of the MinRank matrix code makes a significant difference compared to the Rank Decomposition, as the scheme now avoids relying on the vector $\boldsymbol{x} \in \mathbb{F}_q^k$, leading to a *much smaller* witness. Overall, with the identity in $\boldsymbol{S}$, the inputs are of size

$$(\underbrace{r \cdot (m-r)}_{\boldsymbol{S}} + \underbrace{r \cdot n}_{\boldsymbol{C}}) \cdot \log_2(q).$$

As for RSD, the size does not depend on k anymore, which allows a better selection of the parameters.

Global Comparison. Table 6 provides a global comparison of the different modelings in terms of witness size for the MinRank problem. For each of the described modelings, we provide the size formula as well as the obtained concrete size for optimized parameters reaching a 128-bit security for the attacks described in Sect. 3.3.

Table 6. Modeling for MinRank and resulting witness size in MPC protocols.

Modeling	**Witness size**	**Parameters for $\lambda = 128$**	
		(q, m, n, k, r)	**Size**
Rank Decomposition	$[k + r(m - r) + rn] \cdot \log_2(q)$	$(16, 15, 15, 78, 6)$	111 B
q-polynomial	$[k + rm] \cdot \log_2(q)$	$(16, 15, 15, 78, 6)$	76 B
Kipnis-Shamir	$[k + r(n - r)] \cdot \log_2(q)$	$(16, 15, 15, 78, 6)$	66 B
Dual Support Decomp.	$[r(m - r) + rn] \cdot \log_2(q)$	$(2, 43, 43, 1520, 4)$	41 B

5 The TCitH and VOLEitH Frameworks

The MPCitH paradigm [27] is a versatile method introduced in 2007 to build zero-knowledge proof systems using techniques from secure multi-party computation (MPC). This paradigm has been drastically practically improved in recent years (see, e.g., [3,19,23,28]) and is particularly efficient to build zero-knowledge proofs for small circuits such as those involved in (post-quantum) signature schemes. The MPCitH paradigm can be summarized as follows. The prover emulates "in his head" an ℓ-private MPC protocol with N parties and commits each party's view independently. The verifier then challenges the prover to reveal the views of a random subset of ℓ parties. By the privacy of the MPC protocol, nothing is revealed about the plain input, which implies the zero-knowledge property. On the other hand, a malicious prover needs to cheat for at least one party, which shall be discovered by the verifier with high probability, hence ensuring the soundness property.

In what follows, we describe two recently introduced MPCitH-based frameworks, namely the *VOLE-in-the-Head* (VOLEitH) framework from [12] and the *Threshold-Computation-in-the-Head* (TCitH) framework from [22,23]. We then present the recent optimisations proposed by [11].

5.1 Threshold-Computation-in-the-Head Framework

The TCitH framework has been recently introduced in [22] as an extension of a previous work [23] published at Asiacrypt 2023. While almost all the former MPCitH-based proof system relied on additive sharings, the TCitH framework shows how using Shamir's secret sharings (instead of additives sharings) lead to faster schemes with shorter communication.

We refer the reader to [22,23] for a detailed exposition of the TCitH framework which is only briefly abstracted here. In a nutshell, the TCitH framework relies on MPC protocols with broadcasting, randomness oracle and hint oracle (as previous MPCitH schemes) but using Shamir's secret sharing unlocks the use of non-linear multiparty computation (whereas previous MPCitH schemes are based on linear multiparty computation). More precisely, in the considered MPC protocols, one can compute a sharing $[\![a \cdot b]\!]$ of a product $a \cdot b$ from the sharings $[\![a]\!]$ and $[\![b]\!]$ of the operands by share-wise multiplication (for all i, $[\![a \cdot b]\!]_i \leftarrow [\![a]\!]_i \cdot [\![b]\!]_i$).

The TCitH framework comes with two variants depending on how one commits the input shares: either relying on GGM trees [25] or on Merkle trees [32]. In the present work, we focus on the GGM-tree variant which leads to shorter signature sizes for the considered statements. Moreover, we only consider 1-private Shamir's secret sharings, *i.e.* $\ell = 1$, which gives the best results in our context.

Given some degree-d polynomials $f_1, \ldots, f_m$ from $\mathbb{F}[X_1, \ldots, X_{|w|}]$, we want a zero-knowledge proof of knowledge of a witness w satisfying

$$\forall j \in [1, m], \; f_j(w) = 0.$$

We shall use the proof system TCitH-Π_{PC} described in [22, Section 5.2]. We recall the underlying MPC protocol Π_{PC} in Protocol 1. The sharing $[\![0]\!]$ used in Step 4 of the MPC protocol is a publicly-known degree-1 sharing of zero (for example, $[\![0]\!]_i = \omega_i$ when $e = 0$). This MPC protocol is ℓ-private and sound with false positive probability $\frac{1}{|\mathbb{F}|}$ (see [22, Lemma 2]). In practice, the MPC protocol is repeated ρ times in parallel to achieve a false positive probability of $\frac{1}{|\mathbb{F}|^\rho}$. The soundness error of TCitH-Π_{PC} (when $\ell = 1$) is

$$\epsilon = \frac{1}{|\mathbb{F}|^\rho} + \left(1 - \frac{1}{|\mathbb{F}|^\rho}\right) \cdot \frac{d}{N} \; .$$

1. The parties receive a sharing $[\![w]\!]$, with $\deg[\![w]\!] = 1$.
2. The parties get a uniformly-random degree-$(d-1)$ sharing $[\![v]\!]$ of a random value $v \in \mathbb{F}$ from O_H.
3. The parties receive random values $\gamma_1, \ldots, \gamma_m \in \mathbb{F}$ from O_R.
4. The parties locally compute
$$[\![\alpha]\!] = [\![v]\!] \cdot [\![0]\!] + \sum_{j=1}^{m} \gamma_j \cdot f_j([\![w]\!]) \; .$$
5. The parties open $[\![\alpha]\!]$ to publicly recompute α.
6. The parties output Accept if and only if $\alpha = 0$.

Protocol 1: Π_{PC} – Verification of polynomial constraints. O_R is an oracle which provides public trusted randomness to the parties: in a MPCitH setting, this randomness is provided by the verifier. O_H is an oracle which provides sharings of untrusted values named hints: in a MPCitH setting, these sharings are provided by the prover.

To obtain a signature scheme, we first transform the above MPC protocol into a proof of knowledge (PoK) of soundness error ϵ by applying the TCitH transform. We then perform τ parallel repetitions of this PoK and apply the Fiat-Shamir transform [24]. To achieve a λ-bit security, we take the number ρ of MPC repetitions such that $\frac{1}{|\mathbb{F}|^\rho} \leq 2^{-\lambda}$ and the number τ of PoK repetitions such that $\left(\frac{d}{N}\right)^\tau \leq 2^{-\lambda}$.

The proof transcript (*i.e.* the signature) includes:

- The opened shares $[\![w]\!]_I$ of the witness $w \in \mathbb{F}^{|w|}$, for each of the τ PoK repetitions. In practice, the sent values are the auxiliary values Δw.
- The opened shares of $[\![v]\!]_I$: because v is uniformly-sampled, these shares are communication-free since we rely on the TCitH-GGM variant.
- The degree-d sharing $[\![\alpha]\!]$, for each of the ρ MPC repetitions of the τ PoK repetitions. Since $[\![\alpha]\!]_I$ can be recomputed by the verifier and since the α should be zero, the prover just needs to send $(d+1)-1-1=(d-1)$ shares.
- The sibling paths in the GGM trees, together with the unopened seed commitments.

Moreover, the signature includes a 2λ-bit salt and a 2λ-bit commitment digest that correspond to the last verifier challenge (in the Fiat-Shamir heuristic). Therefore, the signature size when using the TCitH framework in the above setting is (in bits):

$$\textsc{Size}_{\textsc{TCitH}} = 4\lambda + \tau \cdot \left(\underbrace{|w| \cdot \log_2 |\mathbb{F}|}_{[\![w]\!]_I} + \underbrace{(d-1)\cdot \rho \cdot \log_2 |\mathbb{F}|}_{[\![\alpha]\!]} + \underbrace{\lambda \cdot \log_2 N}_{\text{GGM tree}} + 2\lambda \right).$$

5.2 VOLE-in-the-Head Framework

The VOLEitH framework has been introduced at Crypto 2023 [12]. This work provides a way to compile any zero-knowledge protocol in the VOLE-hybrid model into a publicly verifiable protocol. While it has not been introduced as a MPCitH construction, it can yet be interpreted as such. Specifically, [22] shows that the VOLEitH framework can be described in the TCitH syntax. Indeed, this framework is similar to the TCitH framework with $\ell = 1$ and GGM trees, up to several details:

- The secret is stored at $P(\infty)$ when sharing, meaning that $e = \infty$. As a result, to share a value v, one samples a random value r and builds the Shamir's polynomial P as $P(X) := vX + r$. While multiplying two Shamir's sharings when $e = \infty$ is similar than when $e \neq \infty$, the addition operation is slightly different: to add two Shamir's sharings $[\![a]\!]$ and $[\![b]\!]$ of degrees respectively d_1 and d_2 (such that $d_1 \leq d_2$) when $e = \infty$, the parties can compute the following d_2-degree sharing

$$\forall i,\ [\![a+b]\!]_i \leftarrow [\![a]\!]_i \cdot \omega_i^{d_2-d_1} + [\![b]\!]_i,$$

where ω_i is the evaluation point of the ith party.
- The VOLEitH framework relies on a *large field embedding*: in the commitment phase, the prover commits τ N-sharings $[\![w]\!]^{(1)}, \ldots, [\![w]\!]^{(\tau)}$ of the witness w. In the basic TCitH framework, the prover runs τ MPC protocols in parallel, each of them on a different sharing $[\![w]\!]^{(j)}$. In the VOLEitH framework, these N sharings are merged to obtain a N^τ-sharing $[\![w]\!]^{(\phi)}$ living in a large field

extension $\mathbb{K}$ such that the extension degree $[\mathbb{K} : \mathbb{F}]$ is ρ, then the prover runs a unique MPC protocol which takes as input this N^τ-sharing. More precisely, the i^{th} share of $[\![w]\!]^{(\phi)}$ is computed as

$$[\![w]\!]^{(\phi)}_i \leftarrow \phi\left([\![w]\!]^{(1)}_{i_1}, \ldots, [\![w]\!]^{(\tau)}_{i_\tau}\right)$$

where $i_1, \ldots, i_\tau \in [1, N]$ satisfy $(i-1) = (i_1-1)+(i_2-1)\cdot N+\ldots+(i_\tau-1)\cdot N^{\tau-1}$ and ϕ is an one-to-one ring homomorphism between $\mathbb{F}^\tau$ and $\mathbb{K}$ ($\rho \geq \tau$). If the sharings $[\![w]\!]^{(1)}, \ldots, [\![w]\!]^{(\tau)}$ encode the *same* witness w, then we get that $[\![w]\!]^{(\phi)}$ is a valid Shamir's secret sharing of w for which the evaluation point of the i^{th} party is $\phi(\omega_{i_1}, \ldots, \omega_{i_\tau})$ (with ω_i the i^{th} party evaluation point in the standard TCitH setting). The main advantage of this large field embedding is that the resulting soundness error of the proof system is $\frac{d}{N^\tau}$ instead of being $\left(\frac{d}{N}\right)^\tau$ (up to the false positive probability).

– The above optimisation requires that the prover ensures that the τ sharings encode the same value (without revealing this value). To ensure this property, the VOLEitH framework introduces an additional prover-verifier pair of rounds. After committing the input shares (including the hint sharings),
  - the prover commits τ additional uniformly-random sharings $[\![u]\!]^{(1)}, \ldots, [\![u]\!]^{(\tau)}$ of the *same* random value $u \in \mathbb{F}^{\rho+B}$, for $B \geq 0$ an additional parameter,
  - the verifier sends a challenge $(H_1|H_2) \in \mathbb{F}^{(\rho+B)\times(n+\rho)}$,
  - for all $j \in [1, \tau]$, the prover reveals the *digest sharing* $[\![\alpha']\!]^{(j)} := H_1[\![w]\!]^{(j)} + H_2[\![v]\!]^{(j)} + [\![u]\!]^{(j)}$, where $\alpha' \in \mathbb{F}^{\rho+B}$.

  The idea behind this process is that the prover computes the digests of all the plain values encoded in $[\![w]\!]^{(1)}, \ldots, [\![w]\!]^{(\tau)}$ (and in $[\![v]\!]^{(1)}, \ldots, [\![v]\!]^{(\tau)}$) and compares them. If $([\![w]\!]^{(i)}, [\![v]\!]^{(i)})$ and $([\![w]\!]^{(j)}, [\![v]\!]^{(j)})$ encode different values, then their digests $[\![\alpha']\!]^{(i)}$ and $[\![\alpha']\!]^{(j)}$ will differ with high probability. In practice, the parameters ρ and B are chosen such that the probability that two different plain values lead to the same digest is negligible. We further note that taking $(H_1|H_2)$ uniformly at random gives the smallest probability but requires to perform matrix-vector multiplications. Other strategies are possible for $(H_1|H_2)$ such as relying on a polynomial-based hash: this increases a bit the collision probability (so one needs to increase B to compensate) but lightens the computation. This strategy is used in the FAEST signature scheme [13].

We use the VOLEitH framework with the same MPC protocol than with the TCitH framework, namely the MPC protocol Π_{PC} described in Protocol 1, which is equivalent to the QuickSilver VOLE-based protocol [39] in the VOLE setting. The publicly-known degree-1 sharing $[\![0]\!]$ in Protocol 1 when $e = \infty$ can be built as $[\![0]\!]_i = 1$ for all i.

To achieve a PoK with λ-bit security (i.e. $2^{-\lambda}$ soundness error), we take the field extension $\mathbb{K}$ of degree ρ such that $\frac{1}{|\mathbb{F}|^\rho} \leq 2^{-\lambda}$, the number τ of sharings $[\![w]\!]^{(j)}$ such that $\frac{d}{N^\tau} \leq 2^{-\lambda}$ and the additional parameter B such[3] that $B \cdot \log_2 |\mathbb{F}| \geq 16$

[3] As explained previously, the parameter B aims to compensate the security loss due to the use of a polynomial-based hash. Such a hash consists in evaluating in a large

(the latter choice corresponds to the choice in the specification of FAEST [13]). Then we obtain a signature scheme by applying the Fiat-Shamir transform [24] as previously.

The proof transcript (*i.e.* the signature) includes:

– The opened shares $[\![w]\!]_I$ of the witness $w \in \mathbb{F}^{|w|}$. In practice, one sends the auxiliary values of the sub-sharings $[\![w]\!]^{(1)}, \dots, [\![w]\!]^{(\tau)}$.
– The opened shares of $[\![v]\!]_I$. When v is uniformly-sampled, the shares are usually communication-free. However, we need τ sub-sharings of the same (uniformly-random) value v. While the first sharing is communication-free, the $\tau - 1$ others require an auxiliary value to ensure that all the sub-sharings encode the same value.
– The degree-d sharing $[\![\alpha]\!]$, for the single MPC execution. Since $[\![\alpha]\!]_I$ can be recomputed by the verifier and since the α should be zero, the prover just needs to send $(d+1) - 1 - 1 = d - 1$ shares.
– The sibling paths in the GGM trees, together with the unopened seed commitments.
– The opened shares $[\![u]\!]_I$. As for $[\![v]\!]_I$, since all the τ sub-sharings must encode the same random value u, only the first sharing is communication-free and the $\tau - 1$ others require an auxiliary value.
– The degree-1 sharings $[\![\alpha']\!]^{(1)}, \dots, [\![\alpha']\!]^{(\tau)}$. Since the plaintext value α' is the same for all these sharings and since $[\![\alpha]\!]_I^{(j)}$ can be recomputed by the verifier for all j, sending all these sharings costs only $(\rho + B)$ field elements.

Moreover, the signature includes a 2λ-bit salt and a 2λ-bit commitment digest that correspond to the last verifier challenge (in the Fiat-Shamir heuristic). Therefore, the signature size when using the VOLEitH framework is (in bits):

$$\begin{aligned}\textsc{Size}_{\text{VOLEitH}} = \\ & 4\lambda + \tau \cdot \left(\underbrace{|w| \cdot \log_2 |\mathbb{F}|}_{[\![w]\!]_I} + \underbrace{\lambda \cdot \log_2 N}_{\text{GGM tree}} + 2\lambda \right) + \underbrace{(d-1)\rho \cdot \log_2 |\mathbb{F}|}_{[\![\alpha]\!]} \\ & + (\tau - 1) \cdot \left(\underbrace{(d-1)\rho \cdot \log_2 |\mathbb{F}|}_{[\![v]\!]_I} + \underbrace{(\rho + B) \log_2 |\mathbb{F}|}_{[\![u]\!]_I} \right) + \underbrace{(\rho + B) \cdot \log_2 |\mathbb{F}|}_{[\![\alpha']\!]}.\end{aligned}$$

5.3 Additional MPCitH Optimisations

New generic optimizations for MPCitH-based schemes relying on GGM trees have been proposed in a recent work [11]. The improvements are threefold:

domain the polynomial which has the hashed values as coefficients. Thanks to the Schwartz-Zippel lemma, we get that the security loss is of a factor $n + \rho$ (which is the length of the hashed vector). By taking $B \cdot \log_2 |\mathbb{F}| \geq 16$ as in the specification of FAEST, we can securely hash vectors of length at most 2^{16}.

1. Instead of considering τ independent GGM trees of N leaves in parallel, the authors propose to rely on a unique large GGM tree of $\tau \cdot N$ leaves where the i^{th} share of the e^{th} PoK repetition is associated to the $(e \cdot N + i)^{\text{th}}$ leaf of the large GGM tree. As explained in [11], "opening all but τ leaves of the big tree is more efficient than opening all but one leaf in each of the τ smaller trees, because with high probability some of the active paths in the tree will merge relatively close to the leaves, which reduces the number of internal nodes that need to be revealed."
2. The authors further propose to improve the previous approach using the principle of *grinding*. When the last Fiat-Shamir challenge is such that the number of revealed nodes in the revealed sibling paths exceed a threshold T_{open}, the signer rejects the challenge and recompute the hash with an incremented counter. This process is done until the number of revealed nodes is $\leq T_{\text{open}}$. For example, if we consider $N = 256$ and $\tau = 16$, the number of revealed nodes is smaller than (or equal to) $T_{\text{open}} := 110$ with probability ≈ 0.2. The selected value of T_{open} induces a rejection probability $p_{\text{rej}} = 1 - 1/\theta$, for some $\theta \in (0, \infty)$, and the signer hence needs to perform an average of θ hash computations for the challenge (instead of 1). While this strategy decreases the challenge space by a factor θ, it does not change the average number of hashes that must be computed to succeed an attack (since the latter is multiplied by θ). As noticed by the authors of [11], this strategy can be thought of as loosing $\log_2 \theta$ bit of security (because of a smaller challenge space) which are regained thanks to a proof-of-work (performing an average of θ hash computations before getting a valid challenge).
3. Finally, [11] proposes to add another explicit proof-of-work to the Fiat-Shamir hash computation of the last challenge. The signer must get a hash digest for which the w last bits are zero, for w a parameter of the scheme. The same counter as for the previous improvement is used as a nonce in this hash and increased until the w-zeros property is satisfied. This strategy increases the cost of hashing the last challenge by a factor 2^w and hence increases the security of w bits. This thus allows to take smaller parameters (N, τ) for the large tree, namely parameters achieving $\lambda - w$ bits of security instead of λ.

While the authors of [11] focus on VOLEitH, the same optimisations also apply to TCitH. In summary, for a given w, one picks parameters (N, τ) ensuring $\lambda - w$ bits of security. Then fixing T_{open} for these (N, τ) yields a rejection probability $p_{\text{rej}} = 1 - 1/\theta$. The gain in size comes from the smaller parameters (N, τ) on the one hand, and the smaller sibling paths (of size $\leq T_{\text{open}}$ instead of $\approx \tau \log_2 N$) on the other hand. This gain in size is traded for an increased number of Fiat-Shamir hash attempts ($\theta \cdot 2^w$ on average instead of 1).

6 New Signatures Based on $\mathsf{RSD_s}$ and $\mathsf{MinRank}$

In this section, we propose new signature schemes based on the rank syndrome decoding problem and on the MinRank problem. To proceed, we rely on the TCitH and VOLEitH frameworks to obtain non-interactive zero-knowledge proofs of knowledge for these two problems using the new Dual Support Decomposition model described in Sect. 4 and we use the recent MPCitH optimisation presented in Sect. 5.3. Moreover, to have more granularity in the choice of the parameters, we consider that the τ emulations of the MPC protocol might not involve the same number of parties: there will be τ_1 emulations with N_1 parties and $\tau_2 := \tau - \tau_1$ emulations with N_2 parties. The schemes are then secure if $\left(\frac{d}{N_1}\right)^{\tau_1} \cdot \left(\frac{d}{N_2}\right)^{\tau_2}$ for TCitH and $\frac{d}{N_1^{\tau_1} \cdot N_2^{\tau_2}}$ for VOLEitH are negligible (instead of simply $\left(\frac{d}{N}\right)^{\tau}$ and $\frac{d}{N^\tau}$).

6.1 New Signatures Based on RSD_s

The TCitH and VOLEitH frameworks enable us to prove the knowledge of a witness that satisfies some polynomial constraints. In order to get a signature scheme based on the rank syndrome decoding problem, one just needs to exhibit the polynomial constraints which is satisfied by a rank syndrome decoding solution. As shown in Sect. 4.1, solving an RSD_s instance for $\boldsymbol{y}$ and $\boldsymbol{H}$ is equivalent to finding $\boldsymbol{s} = (x_2, \ldots, x_r)$ where $x_i \in \mathbb{F}_{q^m}$ for $i \in \{2, \ldots, r\}$ and $\boldsymbol{C} \in \mathbb{F}_q^{r\times(n-r)}$ such that

$$\boldsymbol{x}\boldsymbol{H}^T - \boldsymbol{y} = \boldsymbol{0} \quad \text{with} \quad \boldsymbol{x} := \begin{pmatrix}1 \ \boldsymbol{s}\end{pmatrix} \cdot \begin{pmatrix}\boldsymbol{I_r} \ \boldsymbol{C}\end{pmatrix} \in \mathbb{F}_{q^m}^n \tag{1}$$

Equation 1 directly gives degree-2 polynomial constraints into the coefficients of $\boldsymbol{s}$ and $\boldsymbol{C}$. Let us assume that the $\boldsymbol{H}$ is in standard form, meaning it can be written as $\boldsymbol{H} = \begin{pmatrix}\boldsymbol{I_{n-k}} \ \boldsymbol{H'}\end{pmatrix}$, where $\boldsymbol{H'} \in \mathbb{F}_{q^m}^{(n-k)\times k}$. Given the inputs $[\![\boldsymbol{s}]\!]$ and $[\![\boldsymbol{C}]\!]$, the hint $[\![\boldsymbol{v}]\!]$ with $\boldsymbol{v} \in \mathbb{F}_{q^m}^{\rho}$ and the MPC randomness $\boldsymbol{\Gamma} = (\gamma_{i,j})_{i,j} \in \mathbb{F}_{q^m}^{(n-k)\times\rho}$, the emulated MPC protocol (repeated ρ times) described in Protocol 1 thus consists of computing

$$[\![\boldsymbol{\alpha}]\!] \leftarrow [\![\boldsymbol{v}]\!] \cdot [\![0]\!] + ([\![\boldsymbol{x_A}]\!] + [\![\boldsymbol{x_B}]\!]\boldsymbol{H'}^T - \boldsymbol{y})\boldsymbol{\Gamma}$$

where $[\![\boldsymbol{x_A}]\!]$ and $[\![\boldsymbol{x_B}]\!]$ are built such as $[\![\begin{pmatrix}\boldsymbol{x_A} \ \boldsymbol{x_B}\end{pmatrix}]\!] = \begin{pmatrix}1 \ [\![\boldsymbol{s}]\!]\end{pmatrix} \cdot \begin{pmatrix}\boldsymbol{I_r} \ [\![\boldsymbol{C}]\!]\end{pmatrix}$.

Signature Size. According to Sect. 5, the signature size using the TCitH framework is (in bits):

$$\textsc{Size}_{\textsc{TCitH}} = 4\lambda + \underbrace{\lambda \cdot T_{\text{open}}}_{\text{GGM tree}} + \tau \cdot \left(\underbrace{|w| \cdot \log_2 q}_{[\![\boldsymbol{s}]\!]_I, [\![\boldsymbol{C}]\!]_I} + \underbrace{(d-1) \cdot \rho \cdot \log_2 q}_{[\![\alpha]\!]} + 2\lambda \right),$$

while the signature size using the VOLEitH framework is (in bits):

$$\begin{aligned}\textsc{Size}_{\textsc{VOLEitH}} = & 4\lambda + \underbrace{\lambda \cdot T_{\text{open}}}_{\text{GGM tree}} + \tau \cdot \left(\underbrace{|w| \cdot \log_2 q}_{[\![\boldsymbol{s}]\!]_I, [\![\boldsymbol{C}]\!]_I} + 2\lambda \right) + \underbrace{(d-1) \cdot \rho \cdot \log_2 q}_{[\![\alpha]\!]} \\ & + (\tau - 1) \cdot \left(\underbrace{\rho \cdot \log_2 q}_{[\![v]\!]_I} + \underbrace{(\rho + B) \log_2 q}_{[\![u]\!]_I} \right) + \underbrace{(\rho + B) \cdot \log_2 q}_{[\![\alpha']\!]},\end{aligned}$$

where $|w| := (r-1)m + r(n-r)$.

Computational Cost. The running time of the signing algorithm can be split in three main parts:

1. The generation of the input shares using seed trees and their commitment. The computational cost scales linearly with the number of input shares. When there are τ_1 MPC emulations with N_1 parties and τ_2 MPC emulations with N_2 parties, the total number of input shares is $\tau_1 \cdot N_1 + \tau_2 + N_2$.
2. The MPC emulation. This step consists in computing the degree-2 broadcast sharing $[\![\boldsymbol{\alpha}]\!]$, knowing that $\boldsymbol{\alpha} = 0$. Let us estimate the cost of emulating the MPC protocol. We only count multiplications which are predominant (compared to additions) for the considered extension fields. We recall that multiplying two degree-1 sharings costs 2 multiplications in the underlying field, assuming we already know the plain value.
   - With TCitH, the MPC emulation will be repeated $\tau := \tau_1 + \tau_2$ times. Each repetition includes 2 vector-matrix multiplications with a matrix $\mathbb{F}_{q^m}^{(r-1)\times(n-r)}$ to compute $[\![\boldsymbol{x}]\!] := [\![(\boldsymbol{x_A}\ \boldsymbol{x_B})]\!]$, 2 vector-matrix multiplications with a matrix of $\mathbb{F}_{q^m}^{k\times(n-k)}$ to compute $[\![\boldsymbol{r}]\!] := [\![\boldsymbol{x_A}]\!] + [\![\boldsymbol{x_B}]\!]\boldsymbol{H'}^T - \boldsymbol{y}$, and 2 vector-matrix multiplications with matrix of $\mathbb{F}_{q^m}^{(n-k)\times\rho}$ to compute $[\![\boldsymbol{\alpha}]\!]$.
   - With VOLEitH, the MPC emulation is executed only once, but in a larger extension field $\mathbb{K}$ where $[\mathbb{K} : \mathbb{F}_{q^m}] = \rho$. The emulation includes 2 vector-matrix multiplications with a matrix $\mathbb{K}^{(r-1)\times(n-r)}$ to compute $[\![\boldsymbol{x}]\!] := [\![(\boldsymbol{x_A}\ \boldsymbol{x_B})]\!]$, 2ρ vector-matrix multiplications with a matrix of $\mathbb{F}_{q^m}^{k\times(n-k)}$ to compute $[\![\boldsymbol{r}]\!] := [\![\boldsymbol{x_A}]\!] + [\![\boldsymbol{x_B}]\!]\boldsymbol{H'}^T - \boldsymbol{y}$, and 2 vector-matrix multiplications with matrix of $\mathbb{K}^{(n-k)\times 1}$ to compute $[\![\boldsymbol{\alpha}]\!]$.
3. The global proof-of-work, composed of the grinding process on the seed trees and the explicit proof-of-work on the Fiat-Shamir hash computation. Its average cost is $\theta \cdot 2^w$ Fiat-Shamir hash computations.

The running time of the other parts of the signing algorithm is negligible compared to those three components. Regarding the running time of the verification algorithm, since the verifier should also expand the seed trees and emulate some parties, the verification time will be similar (a bit smaller) than the signing time.

Parameter Selection. We select some parameter sets for our signature schemes. To have a fair comparison between both frameworks (TCitH and VOLEitH), we chose the parameters such that the cost of generating the input shares and the cost of the proof-of-work are similar (namely, we chose parameters such that $\tau_2 \cdot N_1 + \tau_2 \cdot \tau_2$ and $\theta \cdot 2^w$ are roughly equal). We present in Table 7 the sizes obtained for the signature scheme.

While proposing optimized implementations of our signature scheme is left for future work, we provide some (upper bound) estimates for its running time in Table 8. The timings of the symmetric components (generation and commitment of the input shares and proof of work) are estimated based on the benchmarks from [11]. Since we rely on the same parameters for the symmetric components (same $\tau_1 \cdot N_1 + \tau_2 \cdot N_2$ and same $\log_2 \theta + w$), we can use their timings as upper bounds. For example, their scheme MandaRain-3-128 s includes a generation and commitment of 22 528 input shares and has a total proof-of-work of 14 bits as our "short" instances. Since it runs in 2.8 ms on a 5 GHz CPU, we deduce that the symmetric components cost is *at most* 14 Mcycles.[4] Then, we derived and benchmarked a naive implementation of the MPC emulation, which gives us an upper bound for the emulation cost. Despite this pessimistic estimation, the results presented in Table 8 show that our scheme is competitive with the NIST candidate RYDE. In particular, all our variants relying on VOLEitH are faster than RYDE.

Comparison. Table 9 summarizes the state of the art of signature schemes based on RSD. We include in the comparison only short parameters, i.e., with $N = 256$ for MPCitH-based signatures, and $N = 32$ for [15]. We include the schemes of Stern [37] and Véron [38] applied to the rank metric. For 128 bits of security, these two schemes have signature sizes of around 30 kB. These sizes were roughly halved in [21] and [15]. Finally, [20] reduced it below 6 kB and our work achieves sizes below 4 kB.

Resilience Property. One should note that our scheme is highly resilient to hypothetical cryptanalytic progress on RSD_s. Indeed, if we were to take the set of parameters for RSD_s corresponding to NIST III, applied to the proof of knowledge for NIST I, i.e., a security of $\lambda = 192$ for RSD_s and $\lambda = 128$ for the protocol, we would get an increase of only 0.4 kB (for $N = 512$) or 0.3 kB (for $N = 2048$)

Table 7. Parameters and resulting sizes for the new signature scheme based on RSD_s. The used parameters for the rank syndrome decoding problem are those of Table 3.

Security	Trade-off	Framework	Scheme Parameters				Computational Cost			Signature
			τ	(τ_1, N_1)	(τ_2, N_2)	T_{open}	#Leaves	$\log_2 \theta$	w	
NIST I	Short	TCitH	12	$(10, 2^{11})$	$(2, 2^{10})$	111	22528	5.0	9	2 937 B
		VOLEitH	11	$(0, 2^{12})$	$(11, 2^{11})$	99	22528	7.2	7	2 851 B
	Fast	TCitH	20	$(4, 2^8)$	$(16, 2^7)$	113	3072	7.1	3	3 708 B
		VOLEitH	16	$(8, 2^8)$	$(8, 2^7)$	102	3072	2.9	8	3 450 B
NIST III	Short	TCitH	18	$(2, 2^{12})$	$(16, 2^{11})$	174	40960	4.9	9	6 713 B
		VOLEitH	16	$(4, 2^{12})$	$(12, 2^{11})$	162	40960	2.7	12	6 566 B
	Fast	TCitH	30	$(10, 2^8)$	$(20, 2^7)$	178	5120	6.9	1	8 454 B
		VOLEitH	24	$(16, 2^8)$	$(8, 2^7)$	176	5120	0.0	8	8 207 B
NIST V	Short	TCitH	25	$(5, 2^{12})$	$(20, 2^{11})$	245	61440	5.6	0	12 371 B
		VOLEitH	22	$(8, 2^{12})$	$(14, 2^{11})$	248	61440	0.0	6	12 682 B
	Fast	TCitH	39	$(17, 2^8)$	$(22, 2^7)$	247	7168	3.7	4	14 926 B
		VOLEitH	32	$(24, 2^8)$	$(8, 2^7)$	247	7168	0.0	8	14 768 B

[4] In making this consideration, we include the overhead of emulating their MPC protocol to our estimates of the symmetric part.

in the signature size. Namely, we can take a large margin of security for the parameters of $\mathsf{RSD_s}$ at a moderate cost.

6.2 New Signatures Based on MinRank

The TCitH and VOLEitH frameworks enable us to prove the knowledge of a witness that satisfies some polynomial constraints. In order to get a signature scheme based on MinRank, one just needs to exhibit the polynomial constraints which that a MinRank solution should satisfy. As shown in Sect. 4.2, we can use the MinRank Syndrome problem, which asks to find $\boldsymbol{S'} \in \mathbb{F}_q^{(m-r)\times r}$ and $\boldsymbol{C} \in \mathbb{F}_q^{r\times n}$ such that

$$\rho(\boldsymbol{E}) \cdot \boldsymbol{H}^T - \boldsymbol{y} = \boldsymbol{0} \quad \text{with} \quad \boldsymbol{E} := \begin{pmatrix} \boldsymbol{I_r} \\ \boldsymbol{S'} \end{pmatrix} \cdot \boldsymbol{C}, \tag{2}$$

where $\boldsymbol{H} \in \mathbb{F}_q^{(mn-k)\times mn}$ and $\boldsymbol{y} \in \mathbb{F}_q^{mn-k}$. Equation (2) directly gives degree-2 polynomial constraints into the coefficients of $\boldsymbol{S'}$ and $\boldsymbol{C}$. Let us assume that the matrix $\boldsymbol{H}$ is in standard form, meaning it can be written as $\boldsymbol{H} = \begin{pmatrix}\boldsymbol{I_{n\cdot m-k}} & \boldsymbol{H'}\end{pmatrix}$, where $\boldsymbol{H'} \in \mathbb{F}_q^{(n\cdot m-k)\times k}$. Given the inputs $[\![\boldsymbol{S'}]\!]$ and $[\![\boldsymbol{C}]\!]$, the hint $[\![\boldsymbol{v}]\!]$ with $\boldsymbol{v} \in \mathbb{F}_q^{\rho}$ and the MPC randomness $\boldsymbol{\Gamma} = (\gamma_{i,j})_{i,j} \in \mathbb{F}_q^{(n-k)\times\rho}$, the emulated MPC protocol (repeated ρ times) described in Protocol 1 thus consists in computing

$$[\![\boldsymbol{\alpha}]\!] \leftarrow [\![\boldsymbol{v}]\!] \cdot [\![0]\!] + ([\![\boldsymbol{x_A}]\!] + [\![\boldsymbol{x_B}]\!]\boldsymbol{H'}^T - \boldsymbol{y})\boldsymbol{\Gamma}$$

where $[\![\boldsymbol{x_A}]\!]$ and $[\![\boldsymbol{x_B}]\!]$ are built such as $[\![(\boldsymbol{x_A}\ \boldsymbol{x_B})]\!] = \rho\left(\begin{pmatrix} \boldsymbol{I_r} \\ [\![\boldsymbol{S'}]\!] \end{pmatrix} \cdot [\![\boldsymbol{C}]\!]\right)$.

Table 8. Estimation of the signing times of the new signature scheme based on $\mathsf{RSD_s}$ (in mega-cycles).

Security	Trade-off	Framework	Symmetric Part	MPC Emulation			Total	RYDE
			From [11]	$[\![\boldsymbol{x}]\!]$	$[\![\boldsymbol{r}]\!]$	$[\![\boldsymbol{\alpha}]\!]$		
NIST I	Short	TCitH	14	0.38	1.42	0.19	16.0	23.4
		VOLEitH	14	0.43	0.36	0.07	14.9	
	Fast	TCitH	1.8	0.62	2.36	0.24	5.0	5.4
		VOLEitH	1.8	0.43	0.36	0.07	2.7	
NIST III	Short	TCitH	37	3.9	12.8	0.6	54.3	49.6
		VOLEitH	37	1.3	2.1	0.2	40.6	
	Fast	TCitH	4.4	6.5	21.3	1.1	33.3	12.2
		VOLEitH	4.4	1.3	2.1	0.2	8.0	
NIST V	Short	TCitH	45	9.5	24.4	0.9	79.8	94.9
		VOLEitH	45	2.0	2.9	0.2	50.1	
	Fast	TCitH	6.8	14.8	37.8	1.4	60.8	22.7
		VOLEitH	6.8	1.9	2.9	0.17	11.8	

Table 9. Comparison of the signatures relying on RSD, restricting to the schemes using the Fiat-Shamir transform.

RSD Parameters	Scheme	N	M	τ	η	ρ	Signature Size
$q = 2$ $m = 31$ $n = 33$ $k = 15$ $r = 10$	[37]	-	-	219	-	-	33 886 B
	[38]	-	-	219	-	-	28 794 B
	[21]	32	389	28	-	-	14 792 B
	[15]	32	389	28	-	-	12 816 B
	[20] RD	256	-	21	24	-	8 990 B
	[20] LP and [4] ($\mathsf{RSD_s}$)	256	-	20	1	-	5 956 B
$q = 2, m = 53, n = 53$ $k = 45, r = 4$	Our scheme (TCitH)	256	-	20	-	3	3 708 B
	Our scheme (VOLEitH)	256	-	16	-	128	3 450 B

Signature Size. According to Sect. 5, the signature size using the TCitH framework is (in bits):

$$\textsc{Size}_{\textsc{TCitH}} = 4\lambda + \underbrace{\lambda \cdot T_{\text{open}}}_{\text{GGM tree}} + \tau \cdot \left(\underbrace{|w| \cdot \log_2 q}_{[\![\boldsymbol{S'}]\!]_I, [\![\boldsymbol{C}]\!]_I} + \underbrace{(d-1) \cdot \rho \cdot \log_2 q}_{[\![\alpha]\!]} + 2\lambda \right),$$

while the signature size using the VOLEitH framework is (in bits):

$$\begin{aligned} \textsc{Size}_{\textsc{VOLEitH}} =& 4\lambda + \underbrace{\lambda \cdot T_{\text{open}}}_{\text{GGM tree}} + \tau \cdot \left(\underbrace{|w| \cdot \log_2 q}_{[\![\boldsymbol{S'}]\!]_I, [\![\boldsymbol{C}]\!]_I} + 2\lambda \right) + \underbrace{(d-1) \cdot \rho \cdot \log_2 q}_{[\![\alpha]\!]} \\ & + (\tau - 1) \cdot \left(\underbrace{\rho \cdot \log_2 q}_{[\![v]\!]_I} + \underbrace{(\rho + B) \log_2 q}_{[\![u]\!]_I} \right) + \underbrace{(\rho + B) \cdot \log_2 q}_{[\![\alpha']\!]}, \end{aligned}$$

where $|w| := r(m - r) + rn$.

Computational Cost. As in the previous section, the running time of the signing algorithm can be split in three main parts:

1. The generation of the input share using seed trees and their commitment. The computational cost scales linearly with the number of input shares. When there are τ_1 MPC emulations with N_1 parties and τ_2 MPC emulations with N_2 parties, the total number of input shares is $\tau_1 \cdot N_1 + \tau_2 + N_2$.
2. The MPC emulation. This step consists in computing the degree-2 broadcast sharing $[\![\boldsymbol{\alpha}]\!]$, knowing that $\boldsymbol{\alpha} = 0$. Let us estimate the cost of emulating the MPC protocol (while only counting multiplications as above).
   - With TCitH, the MPC emulation will be repeated $\tau := \tau_1 + \tau_2$ times. Each repetition includes 2 multiplications between matrices of $\mathbb{F}_N^{(m-r)\times r}$ and $\mathbb{F}_N^{r\times n}$ to compute $[\![\boldsymbol{x}]\!]$, $2 \cdot [\mathbb{F}_N : \mathbb{F}_q]$ vector-matrix multiplications with a matrix of $\mathbb{F}_q^{k\times(n\cdot m-k)}$ to compute $[\![\boldsymbol{x_A}]\!] + [\![\boldsymbol{x_B}]\!]\boldsymbol{H'}^T - \boldsymbol{y}$, and 2 vector-matrix multiplications with matrix of $\mathbb{F}_N^{(n\cdot m-k)\times \rho}$ to compute $[\![\boldsymbol{\alpha}]\!]$.
   - With VOLEitH, the MPC emulation is executed only once, but in a larger extension field $\mathbb{K}$ where $[\mathbb{K} : \mathbb{F}_N] = \rho$. The emulation includes 2 matrix multiplications of $\mathbb{K}^{(m-r)\times r}$ and $\mathbb{K}^{r\times n}$ to compute $[\![\boldsymbol{x}]\!]$, $2\rho \cdot [\mathbb{F}_N : \mathbb{F}_q]$ vector-matrix multiplications with a matrix of $\mathbb{F}_q^{k\times(n\cdot m-k)}$ to compute $[\![\boldsymbol{x_A}]\!] + [\![\boldsymbol{x_B}]\!]\boldsymbol{H'}^T - \boldsymbol{y}$, and 2 vector-matrix multiplications with matrix of $\mathbb{K}^{(n\cdot m-k)\times 1}$ to compute $[\![\boldsymbol{\alpha}]\!]$.
3. The global proof-of-work, composed of the grinding process on the seed trees and the explicit proof-of-work on the Fiat-Shamir hash computation. Its average cost is $\theta \cdot 2^w$ Fiat-Shamir hash computations.

The running time of the other parts of the signing algorithm is negligible compared to those three components. Regarding the running time of the verification algorithm, since the verifier should also expand the seed trees and emulate some parties, the verification time will be similar (a bit smaller) than the signing time.

Parameter Selection. We select some parameter sets for our signature schemes. To have a fair comparison between both frameworks (TCitH and VOLEitH), we chose the parameters such that the cost of generating the input shares and the cost of the proof-of-work are similar (namely, we chose parameters such that $\tau_2 \cdot N_1 + \tau_2 \cdot \tau_2$ and $\theta \cdot 2^w$ are roughly equal). We present in Table 10 the sizes obtained for the signature scheme.

As previously, we leave optimized implementations for future work and provide (upper bound) estimates of the running time in Table 11 based on the benchmarks from [11] and a naive implementation of the MPC emulation of our scheme. Despite this pessimistic estimation, the results of Table 11 show that our scheme is competitive with the NIST sublmissions MIRA and MiRitH (both applying MPC-in-the-Head to MinRank). In particular, all our variants relying on TCitH are faster than MIRA and the short instances of MiRitH.

Table 10. Parameters and resulting sizes for the new signature scheme based on MinRank. The used parameters for the MinRank problem are those of Table 4.

Security	Trade-off	Framework	Scheme Parameters				Computational Cost			Signature
			τ	(τ_1, N_1)	(τ_2, N_2)	T_{open}	#Leaves	$\log_2 \theta$	w	
NIST I	Short	TCitH	12	$(10, 2^{11})$	$(2, 2^{10})$	111	22528	5.0	9	2 896 B
		VOLEitH	11	$(0, 2^{12})$	$(11, 2^{11})$	99	22528	7.0	7	2 813 B
	Fast	TCitH	20	$(4, 2^{8})$	$(16, 2^{7})$	113	3072	7.0	3	3 640 B
		VOLEitH	16	$(8, 2^{8})$	$(8, 2^{7})$	102	3072	2.8	8	3 396 B
NIST III	Short	TCitH	18	$(2, 2^{12})$	$(16, 2^{11})$	174	40960	5.0	9	6 584 B
		VOLEitH	16	$(4, 2^{12})$	$(12, 2^{11})$	162	40960	2.7	12	6 452 B
	Fast	TCitH	30	$(10, 2^{8})$	$(20, 2^{7})$	178	5120	6.9	1	8 240 B
		VOLEitH	24	$(16, 2^{8})$	$(8, 2^{7})$	176	5120	0.0	8	8 036 B
NIST V	Short	TCitH	25	$(5, 2^{12})$	$(20, 2^{11})$	245	61440	5.6	0	12 149 B
		VOLEitH	22	$(8, 2^{12})$	$(14, 2^{11})$	248	61440	0.0	6	12 486 B
	Fast	TCitH	39	$(17, 2^{8})$	$(22, 2^{7})$	247	7168	3.8	4	14 579 B
		VOLEitH	32	$(24, 2^{8})$	$(8, 2^{7})$	247	7168	0.0	8	14 484 B

Comparison. Table 12 summarizes the state of the art of signature schemes based on MinRank. We include in the comparison only short parameters, i.e., with $N = 256$ for MPCitH-based signatures, and $N = 32$ for [15]. For the MinRank parameters, we use $q = 16, m = 16, n = 16, k = 142, r = 4$. Historically, the first schemes from [18, 35], and [14] obtained signature sizes no less than 26 kB for 128 bits of security. Then, the technique from [15] applied to MinRank achieved ∼10 kB, and [2] reduced it even below 7 kB. The recent work from [20] reduces it below 6 kB, and the MIRA and MiRitH submissions to the NIST have sizes below 6 kB as well. Finally, our work achieves sizes below 4 kB.

Resilience Property. As for our scheme based on RSD_s, our above scheme is highly resilient to hypothetical cryptanalytic progress on MinRank. Indeed, if we were to take the set of parameters for MinRank corresponding to NIST III, applied to the proof of knowledge for NIST I, i.e., a security of $\lambda = 192$ for MinRank and $\lambda = 128$ for the protocol, we would get an increase of only 0.4 kB (for $N = 512$) or 0.3 kB (for $N = 2048$) in the signature size. Namely, we can take a large margin of security for the parameters of MinRank at a moderate cost.

Table 11. Estimation of the running times of the new signature scheme based on MinRank (in mega-cycles).

Security	Trade-off	Framework	Symmetric Part	MPC Emulation			Total	MIRA	MiRitH
			From [11]	$[\![\boldsymbol{x}]\!]$	$[\![\boldsymbol{r}]\!]$	$[\![\boldsymbol{\alpha}]\!]$			
NIST I	Short	TCitH	14	12.6	4.6	4.5	35.7	46.8	76.5
		VOLEitH	14	54.8	1.4	2.7	72.9		
	Fast	TCitH	1.8	3.7	5.1	1.9	12.5	37.4	8.7
		VOLEitH	1.8	54.8	1.4	2.7	60.7		
NIST III	Short	TCitH	37	37.6	22.0	14.4	111.0	119.7	192.9
		VOLEitH	37	217.8	8.2	7.5	270.5		
	Fast	TCitH	4.4	9.8	23.4	5.2	42.8	107.2	22.5
		VOLEitH	4.4	217.8	8.2	7.5	237.9		
NIST V	Short	TCitH	45	82.4	60.2	33.3	220.9	337.7	308.6
		VOLEitH	45	695.2	3.9	19.1	763.2		
	Fast	TCitH	6.8	15.2	61.7	9.7	93.4	322.3	36.4
		VOLEitH	6.8	694.4	14.6	19.1	734.9		

Table 12. Comparison of the signatures relying on MinRank, restricting to the schemes using the Fiat-Shamir transform.

MinRank Parameters	Scheme	N	M	τ	η	ρ	Signature Size
$q = 16$ $m = 16$ $n = 16$ $k = 142$ $r = 4$	[18]	-	-	219	-	-	28 575 B
	[35]	-	-	128	-	-	28 128 B
	[14]	-	256	128	-	-	26 405 B
	[15]	32	389	28	-	-	10 937 B
	[2]	256	-	18	-	-	7 422 B
	[20] RD	256	-	19	9	-	7 122 B
$q = 16, m = 16, n = 16$ $k = 120, r = 5$	[20] LP and MIRA [5]	256	-	18	1	-	5 640 B
$q = 16, m = 15, n = 15$ $k = 78, r = 6$	MiRitH [1]	256	-	19	9	-	5 673 B
$q = 2, m = 43, n = 43$ $k = 1520, r = 4$	Our scheme (TCitH)	256	-	20	-	130	3 640 B
	Our scheme (VOLEitH)	256	-	16	-	128	3 396 B

References

1. Gora Adj, Stefano Barbero, Emanuele Bellini, Andre Esser, Luis Rivera-Zamarripa, Carlo Sanna, Javier Verbel, and Floyd Zweydinger. MiRitH. NIST's Post-Quantum Cryptography Standardization of Additional Digital Signature Schemes Project (Round 1), https://pqc-mirith.org/, 2023.
2. Gora Adj, Luis Rivera-Zamarripa, and Javier Verbel. Minrank in the head. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, *Progress in Cryptology - AFRICACRYPT 2023*, pages 3–27, Cham, 2023. Springer Nature Switzerland.
3. Carlos Aguilar Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The Return of the SDitH. In Carmit Hazay and Martijn Stam, editors, *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*, volume 14008 of *Lecture Notes in Computer Science*, pages 564–596. Springer, 2023.
4. Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vincotte. RYDE. NIST's Post-Quantum Cryptography Standardization of Additional Digital Signature Schemes Project (Round 1), https://pqc-ryde.org/, 2023.
5. Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Matthieu Rivain, and Jean-Pierre Tillich. MIRA. NIST's Post-Quantum Cryptography Standardization of Additional Digital Signature Schemes Project (Round 1), https://pqc-mira.org/, 2023.
6. Nicolas Aragon, Philippe Gaborit, Adrien Hauteville, and Jean-Pierre Tillich. A New Algorithm for Solving the Rank Syndrome Decoding Problem. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 2421–2425, 2018.
7. Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, and Jean-Pierre Tillich. An Algebraic Attack on Rank Metric Code-Based Cryptosystems. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 64–93, Cham, 2020. Springer International Publishing.
8. Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, and Jean-Pierre Tillich. Revisiting algebraic attacks on MinRank and on the rank decoding problem. *Designs, Codes and Cryptography*, 91:3671-3707, 2023.
9. Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier Verbel. Improvements of Algebraic Attacks for Solving the Rank Decoding and MinRank Problems. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020*, pages 507–536, Cham, 2020. Springer International Publishing.
10. Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier Verbel. Improvements of Algebraic Attacks for Solving the Rank Decoding and MinRank Problems. In *Advances in Cryptology – ASIACRYPT 2020*, pages 507–536. Springer International Publishing, 2020.
11. Carsten Baum, Ward Beullens, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. One tree to rule them all: Optimizing ggm trees and owfs for post-quantum signatures. Cryptology ePrint Archive, Paper 2024/490, 2024. https://eprint.iacr.org/2024/490.

12. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 581–615, Cham, 2023. Springer Nature Switzerland.
13. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST. NIST's Post-Quantum Cryptography Standardization of Additional Digital Signature Schemes Project (Round 1), https://faest.info/, 2023.
14. Emanuele Bellini, Andre Esser, Carlo Sanna, and Javier Verbel. Mr-dss - smaller minrank-based (ring-)signatures. In *Post-Quantum Cryptography: 13th International Workshop, PQCrypto 2022, Virtual Event, September 28-30, 2022, Proceedings*, page 144-169, Berlin, Heidelberg, 2022. Springer-Verlag.
15. Loïc Bidoux and Philippe Gaborit. Compact Post-quantum Signatures from Proofs of Knowledge Leveraging Structure for the PKP, SD and RSD Problems. In *Codes, Cryptology and Information Security (C2SI)*, 2023.
16. Loïc Bidoux, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, and Matthieu Rivain. Dual support decomposition in the head: Shorter signatures from rank SD and MinRank. Cryptology ePrint Archive, Paper 2024/541, 2024.
17. Nicolas Courtois. La sécurité des primitives cryptographiques basées sur des problèmes algébriques multivariables mq, ip, minrank, hfe, 2001.
18. Nicolas T. Courtois. Efficient zero-knowledge authentication based on a linear algebra problem minrank. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pages 402–421, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.
19. Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient Zero-knowledge MPCitH-based Arguments. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 3022–3036. ACM, 2021.
20. Thibauld Feneuil. Building MPCitH-based signatures from MQ, MinRank, Rank SD and PKP. In *International Conference on Applied Cryptography and Network Security (ACNS)*, 2024.
21. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Shared permutation for syndrome decoding: new zero-knowledge protocol and code-based signature. *Designs, Codes and Cryptography*, 91:563–608, 2022.
22. Thibauld Feneuil and Matthieu Rivain. Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments. *Cryptology ePrint Archive, Report 2023/1573*, 2023.
23. Thibauld Feneuil and Matthieu Rivain. Threshold Linear Secret Sharing to the Rescue of MPC-in-the-Head. In *International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt)*, 2023.
24. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.
25. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792-807, aug 1986.

26. Louis Goubin and Nicolas T. Courtois. Cryptanalysis of the TTM Cryptosystem. In *International Conference on the Theory and Application of Cryptology and Information Security*, 2000.
27. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, page 21-30, New York, NY, USA, 2007. Association for Computing Machinery.
28. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 525–537. ACM, 2018.
29. Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In *crypto '99*, volume 1666 of *LNCS*, pages 19–30, Santa Barbara, California, USA, August 1999. Springer.
30. P. Loidreau. Properties of codes in rank metric, 2006.
31. Carlos Aguilar Melchior, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Maxime Bros, Alain Couvreur, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, and Gilles Zémor. RQC. NIST's Post-Quantum Cryptography Standardization Process, https://pqc-rqc.org/, 2017.
32. Ralph C. Merkle. A digital signature based on a conventional encryption function. In Carl Pomerance, editor, *Advances in Cryptology — CRYPTO '87*, pages 369–378, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.
33. NIST. Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process, 2022. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf.
34. A. V. Ourivski and T. Johansson. New Technique for Decoding Codes in the Rank Metric and Its Cryptography Applications. *Probl. Inf. Transm.*, 38(3):237-246, jul 2002.
35. Bagus Santoso, Yasuhiko Ikematsu, Shuhei Nakamura, and Takanori Yasuda. Three-pass identification scheme based on minrank problem with half cheating probability, 2022.
36. Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612-613, nov 1979.
37. Jacques Stern. A new identification scheme based on syndrome decoding. In *International Cryptology Conference (CRYPTO)*, 1993.
38. Pascal Véron. Improved Identification Schemes Based on Error-Correcting Codes. *Applicable Algebra in Engineering, Communication and Computing*, 8(1), January 1997.
39. Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 2986-3001, New York, NY, USA, 2021. Association for Computing Machinery.

Non-Interactive Blind Signatures: Post-Quantum and Stronger Security

Foteini Baldimtsi[1], Jiaqi Cheng[2], Rishab Goyal[2], and Aayush Yadav[1](✉)

[1] George Mason University, Fairfax, USA
{foteini,ayadav5}@gmu.edu

[2] University of Wisconsin–Madison, Madison, USA
{jiaqicheng,rishab}@cs.wisc.edu

Abstract. Blind signatures enable a receiver to obtain signatures on messages of its choice without revealing any message to the signer. Round-optimal blind signatures are designed as a two-round interactive protocol between a signer and receiver. Coincidentally, the choice of message is not important in many applications, and is routinely set as a random (unstructured) message by a receiver.

With the goal of designing more efficient blind signatures for such applications, Hanzlik (Eurocrypt '23) introduced a new variant called non-interactive blind signatures (NIBS). These allow a signer to asynchronously generate *partial* signatures for any recipient such that only the intended recipient can extract a *blinded* signature for a *random* message. This bypasses the two-round barrier for traditional blind signatures, yet enables many known applications. Hanzlik provided new practical designs for NIBS from bilinear pairings.

In this work, we propose new enhanced security properties for NIBS as well as provide multiple constructions with varying levels of security and concrete efficiency. We propose a new generic paradigm for NIBS from circuit-private leveled homomorphic encryption achieving optimal-sized signatures (i.e., same as any non-blind signature) at the cost of large public keys. We also investigate concretely efficient NIBS with post-quantum security, satisfying weaker level of privacy as proposed by Hanzlik.

Keywords: blind signatures · non-interactive schemes · post-quantum cryptography

1 Introduction

A blind signature scheme is a special class of digital signatures that allows a receiver to obtain a signature on a message of the receiver's choice without revealing this message to the signer. Typically, blind signatures are implemented as an interactive protocol between the *signer* (holding a secret key sk) and the *receiver* (holding the message m to be signed and the signer's public verification key vk). At the end of their interaction, the receiver should obtain a valid signature σ on m. The protocol needs to be correct (i.e., the output signature can be verified using vk and m) and additionally, it needs to satisfy two security properties: *blindness* (i.e., the signer does not obtain any information with respect to m) and *one-more unforgeability* (i.e., the receiver cannot create more valid signatures beyond the ones received after interacting with the signer).

Blind signatures were first introduced by Chaum [Cha83] and have been used as a main building block in numerous privacy-preserving applications. One of the most prominent applications, and the one originally considered by Chaum [Cha83] is private electronic payments (e-cash). The idea is that in an e-cash system, a bank (a.k.a. the signer) creates electronic coins through the use of blind signatures. Each coin is associated with a unique serial number selected by the user, and the bank's signature on it serves as proof of its validity. The blindness property guarantees that nobody (including the bank) can trace the spending of user coins, while the unforgeability property guarantees that users cannot forge coins. Beyond e-cash, blind signatures have found applications in e-voting systems [CGT06], anonymous credentials/tokens [PZ13,BL13,FHS15,DGS+18], direct anonymous attestation [BCC04] and coin tumblers for cryptocurrencies [HBG16,HAB+17].

Two-move Barrier. The original blind signature scheme by Chaum [Cha83] requires *two-moves* of interaction: the recipient sends the blinded message to the signer and the signer responds with a signature on the blinded message. The recipient locally un-blinds the signature and outputs the final message/signature pair. It is easy to see that blind signatures are impossible unless the recipient and signer both send at least one message. Thus, *two-move* blind signatures are considered *round-optimal* and are particularly interesting for real-world applications. Beyond their optimal communication efficiency, their unforgeability automatically extends to the concurrent setting [Lin08,HKKL07] (i.e. even when the adversary engages in multiple concurrent sessions with the signer). There has been a long line of research on round-optimal blind signatures under different models and assumptions [BNPS02,Bol03,Fis06,GRS+11,SC12,FHS15, FHKS16,Gha17,KNYY21,AKSY22,BLNS23a].

Non-interactive Blind Signatures: Beyond the Two-move Barrier. In a two-move blind signature scheme, the first message originates from the receiver and typically includes a blinded version of the message to be signed. Interestingly though, in many applications, the signed message is selected randomly and does not come from a specific distribution or has a specific structure. Consider e-cash as an example, where the signed message is a random value denoting the e-coin ID, arbitrarily selected by a user. This observation was recently made explicit by Hanzlik [Han23] who put forward a new notion called non-interactive blind signatures (NIBS).

In a NIBS system, each receiver is associated with a public-secret key pair $(\mathsf{pk}_R, \mathsf{sk}_R)$ such that any signer can asynchronously create partial signatures psig, called *presignatures*, given only the receiver's public key pk_R. The receiver can extract a pair of *blinded* signature σ and message μ from presignature psig using its corresponding secret key sk_R. Completeness states that σ is a valid signature for message μ, and can be verified given only signer's verification key vk. The message μ should be an unpredictable message for the receiver and the signer.

In terms of security, a NIBS scheme should satisfy the properties of (one-more) unforgeability and blindness similar to the interactive setting. As pointed out in [Han23], using some secret input from the receiver (i.e. sk_R) during the computation of the final signature is crucial to achieve blindness without incurring two-moves. Additionally, Hanzlik also extended the idea of NIBS to *tagged* non-interactive blind signatures (TNIBS) to resemble the functionality of *partially blind* signatures. That is, signatures that allow for the inclusion of some un-blinded metadata, called the 'tag', along with the blinded signature. For example, the tag could contain application specific public information such as date, time, purpose, etc.

Non-interactive blind signatures could replace traditional blind signatures in any application where the choice of message is not important, and could be set as a random (unstructured) message. This would lead to protocols with minimal communication complexity. Beyond the case of e-cash, (tagged) NIBS can be used to implement anonymous token systems, à la Privacy Pass [DGS+18], and lottery systems—we refer to [Han23] for a longer discussion on applications. Interestingly, if we consider the PKI model, then a recipient never needs to interact with a signer to send its public key. Thus, a signer can issue and publish presignatures for users without ever interacting with them. This property makes NIBS suitable for many modern applications. As an example, consider cryptocurrency airdrops which is a mechanism to gift coins to users (typically used to bootstrap interest in a new coin). A cryptocurrency could distribute coins to registered user public keys by creating and publishing presignatures, and then users could obtain the final signatures (i.e., the actual coins) in a privacy preserving way.

Existing Constructions. [Han23] provided a generic template for building (T)NIBS in the random oracle (RO) model from verifiable random functions, digital signatures, and general purpose dual-mode non-interactive witness indistinguishable proofs. They also designed a practically efficient NIBS scheme from signatures on equivalence classes [HS14,FHS19], and a TNIBS from tag-based equivalence class signatures [HS21]. Both schemes crucially rely on the use of bilinear pairings for instantiating the underlying equivalence-class signature. Moreover, their security proofs are carried out in the generic group model.

This Work. The goal of this work is two-fold: pushing the barriers both on the definitional framework and on constructions. Since NIBS is a newly introduced primitive, our first focus is to examine the definitional framework proposed in [Han23] and to ask the following natural question:

Is the definitional framework proposed by [Han23] *correctly capturing the desired properties?*

As it turns out, the blindness properties defined by [Han23] only capture a weaker level of privacy, which is not adequate for all the applications NIBS was proposed for. Thus, in this work, we revisit the definitional framework and provide blindness definitions that better capture the intended level of privacy and which should be considered the basis for future NIBS development.

On the construction side, the current state-of-the-art is that we do not have efficient post-quantum NIBS, *even with only conjectured security.* This was left as an important open problem in [Han23] and brings us to the following question:

Can we design efficient (tagged) NIBS from post-quantum assumptions?

We answer the above question in the affirmative, and believe this will lead to further progress in the emerging area of practical (round-optimal) blind signature schemes with post-quantum security (see [LNP22a, AKSY22] and references therein). We summarize our contributions below.

1. We address the privacy shortcomings of the existing NIBS definition framework, and present new stronger blindness properties for NIBS. Our study of stronger blindness definitions is motivated by scenarios in which a blindness attacker might be able to get a hold of some presignatures along with their corresponding message-signature pairs.[1] We also provide a new feasibility result by extending the folklore Fischlin's paradigm [Fis06] to the non-interactive setting, while proving security in our stronger corruption model.
2. We propose a new generic paradigm for designing (non-interactive) blind signatures with *optimal-sized signatures* from any circuit-private leveled homomorphic encryption (LHE) scheme. We show that LHE can upgrade any regular (non-blind) signature into a NIBS without increasing the size of the final signature. We believe that our LHE-based template could serve as an alternative to the famous Fischlin's paradigm. With great ongoing research in the realm of (somewhat/leveled) homomorphic encryption, we further believe that our proposed template might lead to alternate approaches to efficient (non-interactive) blind signatures in the future. As we discuss later, our LHE-based construction enables a new interesting tradeoff leading to optimal-sized signature with large public keys and pre-signatures; while a general NIZK-based solution enables the opposite tradeoff.
3. We also design a practical lattice-based (T)NIBS scheme that satisfies a weaker level of privacy, as proposed in [Han23]. As we discuss later, this is still adequate for a subset of applications mentioned in [Han23]. The concrete costs and overhead of our NIBS scheme are similar to that for state-of-the-art interactive (round-optimal) blind signatures from lattices. We introduce, and prove security of our system under, a more robust variant of the one-more-ISIS assumption [AKSY22], that we call *randomized one-more-ISIS* assumption which may be of independent interest. We do preliminary cryptanalysis of our assumption, and show that all known attacks on the one-more-ISIS assumption [AKSY22] are unsuccessful in breaking our variant.

Related Work on Lattice-based Blind Signatures. Recently, many new round-optimal schemes for lattice-based blind signatures have been proposed [LNP22a, dK22, AKSY22, BLNS23a]. Lyubashevsky et al. [LNP22a] use

[1] At a very high level, one could view our definitions providing a stronger CCA-style blindness guarantee, while existing definitions provide only a CPA-style blindness guarantee.

Table 1. Our construction compared with state of the art two-move lattice-based blind signatures. $R \rightarrow S$ communication is 0 bytes if a PKI exists, else it is a one-time cost unlike the two-move schemes that have a linear dependence.

		Communication Complexity		
Scheme	**Assumption**	$R \rightarrow S$	$S \rightarrow R$	$\|\sigma\|$
[AKSY22]	OM-ISIS	0.96 KB	0.56 KB	45 KB
[BLNS23a]	MSIS and MLWE	> 100 KB	∼ 60 KB	22 KB
Our Construction (7.1)	rOM-ISIS	0 B	0.96 KB	68 KB

one-time signatures, and build blind signatures under standard lattice assumptions (MSIS, MLWE). However, their scheme only supports bounded polynomial number of signatures per public key with each signature being around 150 KB in size. This was improved by del Pino and Katsumata [dK22] who adapted Fischlin's paradigm [Fis06] to the lattice setting. The resulting scheme allows for an unbounded number of signatures, of around 102 KB each, and is also secure under standard lattice assumptions. Agrawal et al. [AKSY22] significantly improved both the signature, as well as the transcript size, by also leveraging Fischlin's paradigm, but relying on efficient lattice-based NIZK for linear relations [LNP22b]. Their final signature size is 45 KB and the total transcript is just over 1 KB. The security of their scheme is based on the one-more ISIS assumption [AKSY22]. Most recently, Beullens et al. [BLNS23a] improved the Agrawal et al. scheme by off-loading some inefficient computation to the receiver's first message. This has the twin benefit of simultaneously allowing a reduction to standard lattice assumption, as well as reducing the signature size to just 22 KB. However, this came at the cost of the receiver having to prove the validity of cryptographic hash function input-output pair in its first message. This makes the first message significantly less efficient at a few hundred kilobytes. We compare the concrete cost of our lattice-based NIBS construction with the current state-of-the-art round-optimal blind signatures from lattices in Table 1.

2 Technical Overview

In this section, we provide a high-level technical overview and summarize our main contributions. We start by recalling the notion of non-interactive blind signatures.

2.1 Defining Non-Interactive Blind Signatures

In a NIBS system, the Setup algorithm generates the system's (global) public parameters pp (viewed as CRS in-the-sky). There are two key generation algorithms, $\mathsf{KeyGen}_S \rightarrow (\mathsf{sk}, \mathsf{vk}), \mathsf{KeyGen}_R \rightarrow (\mathsf{sk}_R, \mathsf{pk}_R)$, for the signer and receiver, respectively. Given sk, the signer runs a (randomized) Issue algorithm for any receiver's public key pk_R to compute a presignature, psig and nonce. Here nonce

roughly denotes the randomness used by the signer. The receiver then runs the Obtain algorithm on the above presignature to compute the final message-signature pair (μ, σ) using secret key sk_R[2].

[Han23] also extended the above basic notion to *tagged* NIBS (TNIBS). The goal was to capture the notion of partially blind signatures [AF96] which allow a signer and recipient to jointly agree on a public metadata/value to be included as part of the signed message. This is captured by allowing such a common metadata/value, called a *tag*, to be chosen explicitly by the signer, and shared with the receiver along with the presignature. Syntactically, TNIBS is defined identically to NIBS, except Issue, Obtain, and Verify algorithms take tag τ as an additional input.

The Need for Reusability. An essential property of NIBS is *reusability*, which says that a signer can issue multiple distinct presignatures (leading to multiple distinct message-signature pairs) for the *same* receiver public key pk_R. Hanzlik [Han23] attempted to capture this property by providing a random nonce value as input to the Issue algorithm. Unfortunately, as we explain in Sect. 4, the existing formulation is insufficient in capturing the desired reusability property. The issue is that, under the current formulation, there can exist trivial NIBS scheme where Issue and Obtain algorithms simply ignore nonce, thus lead to a limited one-use system. To fix this, we formalize reusability as its own property. Informally, it says that any receiver should obtain two distinct message-signature pairs for two distinct presignature-nonce pairs for any given receiver. Fortunately, the bilinear-based NIBS schemes [Han23] already satisfy reusability, but just did not prove/define it formally.

NIBS Security. For security, we want NIBS to satisfy one-more unforgeability and blindness. One-more unforgeability for NIBS can be defined as a natural extension of the one-more unforgeability for traditional blind signatures [PS96]: the adversary gets access to a presignature oracle, and after receiving ℓ presignatures for recipient public keys specified by the adversary, the adversary must return $\ell + 1$ valid signatures.

On the other hand, defining blindness for NIBS is much more nuanced than for traditional blind signatures[3]. Intuitively, it is captured by defining unlinkability between presignatures and final signatures. More formally, Hanzlik [Han23] proposes the following experiment: the adversary receives two recipient public keys $\mathsf{pk}_{R_0}, \mathsf{pk}_{R_1}$, and outputs two presignatures $\mathsf{psig}_0, \mathsf{psig}_1$ (one for each). The challenger extracts the final signature-message pairs $(\mu_0, \sigma_0), (\mu_1, \sigma_1)$ from these, and the scheme is said to satisfy *receiver blindness* if the attacker cannot link the final signature-message pairs to the presignatures. Abstractly, receiver blindness could be thought of as an 'inter-receiver' blindness property, since it guarantees that a malicious signer cannot figure out the recipient of a final (blinded) signature between two possible options.

[2] Our syntax is nearly identical to that proposed in [Han23]. We deviate slightly in the handling of nonce as we discuss in this section, and also later in Sect. 4.

[3] Essentially, this is because the receiver does not choose the message in the non-interactive setting.

Hanzlik [Han23] also proposed a secondary blindness notion, called *nonce blindness*. This can be viewed as an *intra-recipient* blindness property, where what we want is that a signer issuing more than one presignature to a *specific* user should not be able to link the final message-signature pairs to the corresponding presignatures. Essentially, this provides some flavor of "ordering" unlinkability. This was formalized by Hanzlik [Han23] via an experiment similar to that of receiver blindness with the difference that the adversary is only given one recipient public key pk_R, while it still outputs two presignatures $\mathsf{psig}_0, \mathsf{psig}_1$. The challenger extracts the final signature-message pairs $(\mu_0, \sigma_0), (\mu_1, \sigma_1)$ from these, and the scheme is said to satisfy *nonce blindness* if the attacker cannot link the final signature-message pairs to the original presignatures.

The Issues with [Han23] **Blindness.** Unfortunately, the two blindness definitions given by Hanzlik [Han23] do not really capture a sufficient level of privacy that would be required in most NIBS applications.

The core issue is that both definitions only guarantee unlinkability of two presignatures with their corresponding message-signature pairs. That is, a NIBS scheme secure under the existing nonce blindness might not satisfy unlinkability when given three (or more) presignatures to different users. To showcase the problem consider a concrete scenario wherein an adversary issues two presignatures to a user with pk_{R_0} and one presignature to a user with pk_{R_1}. Once the adversary learns two signatures of a user (through a verifier), they will know with certainty[4] that this must have been the user with pk_{R_0}.

New Stronger Blindness Definitions. To capture such practical attacks described above, we propose a new stronger security framework for NIBS blindness properties. The motivation is to capture scenarios where an adversary is able to learn some correlation between presignature-nonce pairs and their corresponding blind signature-message pairs. We formulate this by providing an adversary oracle access to the receiver's secret key in the form of Obtain queries. To ensure this is not trivially impossible, we restrict the adversary to not make an Obtain query on any of the two challenge presignatures $\mathsf{psig}_0, \mathsf{psig}_1$.

We call these the strong receiver (resp. nonce) blindness properties. At a high level, these can be viewed as CCA-version of blindness (since adversary gets query access to the secret key like in IND-CCA). These definitions guarantee blindness to hold even when a malicious signer is able to bypass blindness of signatures from previous sessions. Now any NIBS scheme that is secure under the above stronger definitions protects against the attacks described earlier. This is because we are letting an adversary break blindness of all non-challenge presignatures, and still ask that the blindness of the challenge presignatures is unaffected. We discuss our definitions in detail later in Sect. 4.

When is Basic-blindness [Han23] **Sufficient?** There are many applications, as discussed in the introduction as well as [Han23], where the basic-blindness

[4] We note that this is not a problem in regular blind signatures, where blindness is indeed defined as an indistinguishability game. However the creation of user assigned presignatures in NIBS complicates the definition of blindness.

receiver and nonce blindness properties as defined by [Han23] are not sufficient. Thus, one of our main assertions and contributions is that for typical applications of NIBS, where multiple presignatures are issued to multiple recipients, the strong blindness definitions should be used.

We also remark that there are some applications where the notion of basic blindness will be sufficient. As a concrete application, consider one-per user anonymous airdrops or e-voting tokens, where only a single presignature will be generated for each recipient. In such applications the notion of (basic) receiver-blindness alone, as defined by [Han23], is sufficient. More broadly, any NIBS scheme satisfying only basic receiver blindness implies a round-optimal (interactive) blind signature scheme for random messages. Thus, any NIBS scheme with basic blindness already covers all known applications of traditional blind signatures where messages are selected randomly. Therefore, we believe that a NIBS scheme satisfying only receiver-blindness is interesting for some applications. Since achieving basic-blindness is relatively easier than achieving strong-blindness, a NIBS scheme secure as per the basic blindness can lead to schemes with better concrete efficiency.

In this work, we provide new constructions and templates for designing NIBS that are secure as per our stronger blindness definitions. We also design new practically efficient NIBS scheme secure as per the basic blindness definitions [Han23]. This leads to first post-quantum NIBS scheme. We leave the problem of designing practically efficient NIBS with strong blindness as an interesting open problem.

2.2 Extending Fischlin's Paradigm to NIBS

In this section, we start describing our main technical ideas. Our starting point is the well-known Fischlin's paradigm [Fis06] for constructing *traditional* blind signatures. At a high level, Fischlin's scheme assumes a common reference string (CRS), a standard signature scheme S, an encryption scheme $\mathcal{E}$, a commitment scheme COM and general NIZK proofs.

The CRS contains an encryption public key pk_E, while each signer simply generates its key pair $(\mathsf{sk}, \mathsf{vk})$ by running the setup of the signature scheme. The user computes $c = \mathsf{COM}(m)$ and sends c to the signer. The signer runs $\mathsf{Sign}(\mathsf{sk}, c)$ to produce σ' and sends σ' to the user. If the signature verifies, the user computes $\mathsf{ct} = \mathsf{Enc}_{\mathsf{pk}_E}(c \parallel \sigma')$ and outputs ct, m and a NIZK π proving knowledge of signature σ' and message m such that σ' is a signature on a commitment of m. Intuitively, the one-more unforgeability follows from binding of the commitment scheme and unforgeability of the signature scheme, via straight-line extraction enabled by the trapdoor key sk_E corresponding to pk_E. Further, blindness follows from the semantic security of the encryption scheme, zero-knowledge of the NIZK, and hiding property of the commitment. Consider the following natural extension of Fischlin's paradigm to the non-interactive setting.

Adapting Fischlin's Paradigm to NIBS. Given that there is no interaction between the signer and the user, the receiver's public key will, in a sense, replace the commitment submitted in the first move. The challenge now is that the commitment can no longer represent a commitment of a single message but has to be a succinct commitment of an exponential number of messages at once. The hope is that signer can sign the commitment in a way such that the signature obliviously binds to exactly one of those messages at random. This binding to one message from an exponential set is important for one-more unforgeability as otherwise the receiver could cheat. Lastly, the receiver can use NIZKs to prove knowledge of the signature and reveals the choice of message that was obliviously selected.

Thus, to extend the above template, we need to overcome two challenges: (a) how to commit to an exponential number of messages efficiently, and (b) how can the signer obliviously select one of those messages. Our approach is to set the receiver's public key pk_R as a commitment to a PRF key K, and the key K (along with its opening) as the secret key. Note that this implicitly defines the exponential set of messages as all the outputs of the PRF. Then, during the issuance of a presignature, the signer can obliviously sample one of the messages by selecting a random input r and signing it along with pk_R. The user obtains a final signature on the message $m = F_K(r)$ and along with m it outputs σ to be a NIZK proof of knowledge of a (pre)signature on pk_R and r corresponding to the message m.

Given that the PRF is a deterministic function and commitment is binding, our approach guarantees that the receiver can only obtain a single final signature from a presignature which allows us to prove one-more unforgeability. Further, blindness properties can be reduced to the zero-knowledge property of the underlying NIZK and hiding of the commitment scheme. Moreover, as we discuss in detail in the full version, we can prove our scheme to be secure under our stronger blindness definitions. The core intuition is that the NIZK proof systems are multi-theorem zero-knowledge, thus they can be used to simulate responses to all Obtain queries including the challenge queries, thus each blinded signature is completely unlinkable to its presignature, since the blinded signature is simulated without any information about the presignature.

Remark 1. Hanzlik [Han23] proposed a similar generic template for NIBS using verifiable random functions and general-purpose dual mode witness indistinguishable proofs, but only proved security under his restricted blindness notions. Our construction is notably simpler as it only requires a PRF and general-purpose NIZK, and provides stronger blindness security. Please refer the full version for the complete construction satisfying our strong blindness definitions.

2.3 A New Template: Circuit-Private LHE to NIBS

We now describe a new template for designing NIBS with stronger blindness based on circuit-private LHE. We already know that homomorphic encryption with special properties [AJL+12,MW16] are useful in designing round-optimal MPC protocols. Moreover, such protocols can be client-optimized where the communication complexity of a client is extremely low. Our main observation is that we can instantiate a NIBS scheme as a specialized two-round MPC protocol where the first round message can be reused [IKO+11, AMPR14, MR17, BJOV18, CDI+19], and by using FHE to instantiate the protocol, we can optimize the communication complexity for the receiver to nearly optimal. Let us elaborate on our main ideas below.

Our key idea is that rather than using NIZKs to hide the receiver's secrets, we can leverage the fact that leveled homomorphic encryption (LHE) enables arbitrary homomorphic operations on the receiver's commitment. Specifically, the receiver commits to a PRF secret key K by encrypting it under LHE. Using this commitment, a signer issues a presignature as follows: it first homomorphically evaluates the PRF $F_K(\cdot)$ on some randomness r of its choice and then homomorphically generates a signature on $F_K(r)$ treated as a message. The resulting ciphertext $\widehat{\mathsf{ct}}$ is, therefore, an encryption of the signature on $F_K(r)$. Under the mild assumption that the LHE is circuit-private [OPP14], the signer can simply send $\widehat{\mathsf{ct}}$ as the presignature along with an argument of knowledge that $\widehat{\mathsf{ct}}$ was evaluated using the signing key as input to the circuit, and randomness r as the nonce. The receiver sets its message to $F_K(r)$ and obtains the corresponding signature by decrypting $\widehat{\mathsf{ct}}$. Notably, the final signature is just a regular signature and is thus optimal in size. Moreover, beyond optimality, this construction satisfies our stronger notion of blindness: strong receiver and nonce blindness.

At a high level, both strong-receiver/nonce blindness follow from the $\mathsf{IND\text{-}CPA}$ security of the encryption scheme which ensures that ct does not reveal anything about K, and the pseudorandomness of F, which ensures that two messages on different nonces look random. The one-more unforgeability of the protocol follows from the unforgeability of the underlying signature scheme and the circuit-privacy of the LHE. For the proof to go through, we additionally require that both parties also prove knowledge of their secret keys. That is, our proof is in the knowledge of secret key (KOSK) model [MOR01, Bol03]. As a supplementary contribution, we also prove that any NIBS scheme secure in the KOSK model is also secure in the standard model, assuming existence of NIZKs.

Our LHE-based NIBS construction is described in Sect. 5, and the generic compiler to upgrade security in KOSK model to standard model is provided in Appendix A. We highlight that our construction explores a new interesting trade-off that yields optimal-sized signatures while paying in terms of higher setup and computation costs. We believe this could lead to alternate approaches to practical NIBS (and round-optimal interactive blind signatures) in the future.

2.4 Making Fischlin-Based NIBS Practical and Post-Quantum

The next contribution of our work is a practically efficient NIBS scheme that satisfies basic-blindness definitions [Han23]. Moreover, we provide a proof-of-concept estimate of all the parameters sizes of our NIBS construction. Let us start by revisiting the Fischlin-based NIBS construction that we explained earlier in Sect. 2.2. We show how to adapt our construction by combining with new ideas such that it makes the design concretely efficient, yet it satisfies (basic) receiver-blindness.

With the above goal, we look back to the recent work by Agrawal et al. [AKSY22] on desgining practical round-optimal (interactive) lattice-based blind signatures. Agrawal et al. suggested that the usage of NIZKs is somewhat unavoidable when designing round-optimal blind signatures from lattices[5]. Thus, their main insight was that, by optimizing the NP language for which NIZKs are needed, one could instantiate Fischlin's paradigm efficiently. In particular, they showed that NIZKs for linear relations, which are already known to be efficiently implementable from lattice assumptions [LNP22b], are sufficient for building practically efficient round-optimal blind signatures.

Our initial attempt is to follow a similar approach. We start by optimizing our NIZK-based template for designing NIBS, similar to how [AKSY22] optimized Fischlin's paradigm. The challenge here is to remove all inefficient generic cryptographic components currently used in our NIZK relation, and implement them via just linear relations. As a starting point, consider the standard hash-then-sign paradigm. A signer samples lattice trapdoor as $(\mathbf{C}, \mathbf{T_C})$ and outputs $\mathbf{C}$ as verification key vk. Receiver starts by randomly selecting a PRF key K, and sets its public key $\mathsf{pk}_R = \mathsf{Com}(K; s)$ as a commitment of K with its secret key $\mathsf{sk}_R = (K, s)$, for some random string s. Signer then assigns presignatures to receiver: it uniformly samples randomness r as nonce, and provides a short preimage of $\mathsf{H}(\mathsf{pk}_R||r)$ as the presignature, where $\mathbf{C} \cdot \mathbf{z} = \mathsf{H}(\mathsf{pk}_R||r)$ and $\mathsf{psig} = \mathbf{z}$, $\mathsf{nonce} = r$. Finally, receiver generates message μ as $F_K(r)$, and signature π as a NIZK proof establishing that $\mathbf{z} = \mathbf{C}^{-1}(\mathsf{H}(\mathsf{pk}_R||r))$, $\mu = F_K(r)$ and $\mathbf{z}$ is short.

Such a design essentially instantiates our NIBS template with the lattice-based signature scheme in [GPV08]. Observe that the major source of inefficiency in the NIZK arises from the hash evaluation and the PRF evaluation. These evaluations are extremely heavy and thus not very practical to prove in a NIZK. Our strategy is to entirely exclude both computations from the NIZK relation. Following from the blueprint of [AKSY22], it is possible to remove the hash evaluation (from the NIZK relation) if we fix the receiver's public key as $\mathsf{pk}_R = \mathbf{A} \cdot \mathbf{x} + \mathsf{H}(\delta)$ for some randomly sampled $\mathbf{x}$ and hash input δ, where $\mathbf{A}$ is a random matrix part of the CRS. Evaluations of $\mathsf{H}(\delta)$ could then be performed outside of the NIZK proof system, if the signature contains δ in the clear. This removes the need to prove costly hash evaluations using the NIZK. The reason this does not break blindness is that even if the signer learns δ, it still cannot link δ with pk_R as $\mathbf{x}$ contains enough entropy so that $\mathbf{A} \cdot \mathbf{x}$ statistically hides all information about $H(\delta)$.

However, at this point, the parallels between our design and the interactive blind signature scheme in [AKSY22] start to diminish. This is because, in the interactive setting, a receiver can simply select δ as a fresh message in each new session. But, in NIBS, we need to create multiple signatures for the same receiver key pk_R. And, the issue is pk_R binds to a fixed value for the lifetime of the system. Thus, unlike [AKSY22], we cannot set δ as the final signed message. This

[5] This is due to the fact that all existing practical lattice-based signature schemes do not support simple algebraic homomorphisms in a practically efficient way, unlike signatures from pairings or factoring-based assumptions.

is because this will violate reusability, and make the NIBS scheme only single-use. Therefore, it will not be any more advantageous than a regular two-round blind signature scheme. Basically, this suggests that we cannot simply replace the usage of a PRF within our generic template as easily! Moreover, if we keep on using the PRF to generate a fresh message from each distinct presignature computed by the signer, then the receiver will have to prove it using the NIZK which will be extremely inefficient.

In summary, this means we can no longer use δ (inside each receiver's key), or the PRF trick to generate a fresh message from each distinct presignature computed by the signer. To this end, we propose an entirely different approach to generate the final messages from presignatures. Our insight here is that the extracted final messages need *not* be 'truly random messages', but it is enough to have (1) the messages be just uncorrelated amongst different receivers, and (2) be distinct for any two distinct valid presignatures for the same receiver.

Our idea is to set the final message as $\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_\perp \\ \mathbf{z}_\perp \end{bmatrix}$ instead of $F_K(r)$, where $\mathbf{C} \cdot \mathbf{z} = \mathsf{pk}_R$. Here we write $\mathbf{x}_\top$ and $\mathbf{x}_\perp$ to be the top and bottom halves of $\mathbf{x}$ (respectively), and the same notation applies to $\mathbf{z}$. The intuition is that, while the receiver's public key contains the entire $\mathbf{x}$ vector, if we set the parameters appropriately, then we can argue by the leftover hash lemma that $\mathbf{x}_\perp$ is statistically hidden from the signer's view. We point out that it is important in our design that the signer implicitly assigns nonce to be $\mathbf{z}_\perp$ (with presignature $\mathbf{z}_\top$) where $\mathbf{C} \cdot \mathbf{z} = \mathsf{pk}_R$, instead of sampling r as randomness and $\mathbf{z}$ as preimage of $\mathsf{pk}_R || r$. This ensures that NIZKs for linear relations are still sufficient as well as makes the design even more optimal. For ensuring this property from preimage sampling, we rely on the Bonsai trick [CHKP10, ABB10b] for lattice trapdoors. Using the standard Bonsai trick, one can generate a preimage $\mathbf{z}$ with the above special property, i.e. $\mathbf{z}_\perp$ is sampled as a random Gaussian vector with appropriate norm.

The above serves as the core skeleton of our efficient construction in Sect. 7 however, as we discuss next, we need to make a few more slight modifications to ensure that we can prove desired unforgeability and blindness security properties for our NIBS scheme.

2.5 Security and the Randomized OM-ISIS Assumption

Since the lattice-based core of our optimized NIBS scheme shares many similarities with interactive blind signature scheme of [AKSY22], a natural first attempt is to prove unforgeability of our NIBS scheme using the same proof strategy as used in their work. Now, to prove security of their round-optimal blind signature scheme, Agrawal et al. [AKSY22] proposed a new ISIS-like assumption, called the one-more ISIS assumption ($\mathsf{OM\text{-}ISIS}$)[6]. Below we briefly recall the assumption[7].

[6] The "one-more" name inspired from one-more-RSA assumption [BNPS03].

[7] For ease of exposition, we present a simplified assumption here. In the original assumption, the set of target vectors T can be chosen adaptively by the challenger.

1. The challenger samples a challenge matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ along with a large set of random target vectors $T \subset \mathbb{Z}_q^n$. It provides the attacker with $\mathbf{A}$ and T.
2. $\mathcal{A}$ can make preimage queries for any target vector $\widehat{\mathbf{t}} \in \mathbb{Z}_q^n$ to which the challenger replies with a short[8] vector $\widehat{\mathbf{x}}$ such that $\mathbf{A} \cdot \widehat{\mathbf{x}} = \widehat{\mathbf{t}}$.
3. OM-ISIS assumption says that $\mathcal{A}$, having made at most ℓ preimage queries, cannot output $\ell+1$ distinct vector pairs $\{(\mathbf{x}_j, \mathbf{t}_j)\}_{j \in [\ell+1]}$ such that $\mathbf{A} \cdot \mathbf{x}_j = \mathbf{t}_j$, $\mathbf{t}_j \in T$, $\mathbf{x}_j$ is sufficiently short.

Intuitively, the assumption says that an adversary cannot find *good* (i.e., reasonably short) preimages for a set of $\ell + 1$ randomly selected vectors, even when it has access to a preimage sampling algorithm that can generate at most ℓ preimages for adversarially selected target vectors.

Agrawal et al. also performed some preliminary cryptanalysis of their assumption. While they were able to design practical attacks [AKSY22, § 4.5] for certain parameter settings, they suggested that, by carefully selecting the parameters in the above assumption, all known attacks fail. Moreover, they proved that their two-round blind signature scheme can be proven secure under the above assumption with those parameters.

However, the presence of efficient practical attacks on a wide range of parameters on the OM-ISIS assumption suggests that this assumption is not very stable and robust. On a technical level, the assumption appears quite strong since an attacker can ask for short preimages for "*any*" target vector of its choice. In more detail, an attacker can simply ask for multiple short preimages for the all-zeros vector. Given such preimage vectors, an attacker can combine them to create an approximate lattice trapdoor. As Agrawal et al. discussed, the quality of such a trapdoor computed is worse than the actual trapdoor, thus such a trapdoor would be useless if the $\ell + 1$ preimage vectors that the attacker must compute have to be as short as the preimage vectors it received.

Unfortunately, setting the parameters carefully sidesteps just one limitation of the OM-ISIS assumption, but does not make the assumption truly robust. Simply put, we believe that providing an unrestricted preimage query access to the attacker is too strong. To further illustrate this, consider an even simpler attack that finds short preimage vectors without creating a lattice trapdoor. The attacker just makes two preimage queries (which correspond to preimage queries)—one on vector $\mathbf{0}$ and other on any non-zero vector $\mathbf{t} \in T$ (T is the target set). Let $\mathbf{z}_1$ and $\mathbf{z}_2$ be the respective preimage vectors. That is, $\mathbf{A} \cdot \mathbf{z}_1 = \mathbf{0}$ and $\mathbf{A} \cdot \mathbf{z}_2 = \mathbf{t}$. Given $\mathbf{z}_1$ and $\mathbf{z}_2$, an attacker simply output three distinct vectors $\mathbf{z}_2$, $\mathbf{z}_2 - \mathbf{z}_1$, and $\mathbf{z}_2 + \mathbf{z}_1$ as the preimages for the same target vector $\mathbf{t} \in T$.

Abstractly, the issue is that an adversary can perform simple linear combinations on preimage vectors, and such linear combinations map to appropriate linear combination of the corresponding target vectors. While this does not qualify as an attack on the two-round blind signature scheme of Agrawal et al. [AKSY22] (since any admissible attacker in their system must be generating preimages on

[8] As in typical lattice settings, by short vectors we mean vectors with small norms.

$\ell+1$ distinct vectors), this highlights that the $\mathsf{OM\text{-}ISIS}$ assumption is susceptible to arbitrary linear combination attacks.

Existence of such linear combination attack strategies are a big barrier to designing reusable non-interactive blind signatures. Our initial attempt to build NIBS as a generalization of the Agrawal et al. scheme turns out to be insecure. The attack is basically the same as the one described above. To ensure reusability, we set the final message as $\mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_\perp \\ \mathbf{z}_\perp \end{bmatrix}$ rather than just δ (where δ is was used in the receiver's public key). An attacker simply makes one presignature query for some receiver public key pk_R, and one presignature query for the all-zeros vector (as the public key), and combines them linearly to obtain > 2 valid presignatures for pk_R. One can easily show that the resulting final messages for all these preimage vectors will also be different, thereby constituting an efficient attack on one-more unforgeability of our basic NIBS scheme.

Randomized OM-ISIS Assumption. To thwart the aforementioned linear-combination style attacks on the OM-ISIS assumption, we propose a new variant that we call *randomized one-more ISIS* assumption $\mathsf{rOM\text{-}ISIS}$. Our goal here is twofold— (a) we want to turn OM-ISIS assumption into a more robust assumption such that there do not exist any efficient/practical attacks (irrespective of how the parameters are set), (b) we can design a non-interactive (as well as two-round) blind signature scheme which can be proven under the new assumption.

Our strategy is to prevent an attacker from learning preimages on arbitrary target vectors of its choice. This was the central property that was exploited by Agrawal et al. [AKSY22] in their practical attacks/cryptanalytic efforts. To this end, we make the challenger "re-randomize" each target vector (independently for each query) before computing its preimage. In turn, this takes away the attacker's prior advantage from obtaining distinct short preimages for the same target vector (or any target of its choice more generally). Moroever, by carefully selecting how the per-query re-randomization happens, we can also avoid all known affine attacks that we discussed. Below we summarize our new randomized one-more ISIS assumption $\mathsf{rOM\text{-}ISIS}$.

1. The challenger samples a challenge matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\underline{\text{a randomization matrix } \mathbf{B} \in \mathbb{Z}_q^{n\times m}}$ along with a large set of random target vectors $T \subset \mathbb{Z}_q^n$. It provides the attacker with $\mathbf{A}$, $\underline{\mathbf{B}}$ and the vector set T.
2. $\mathcal{A}$ can make preimage queries for any target vector $\widehat{\mathbf{t}} \in \mathbb{Z}_q^n$ such that the challenger replies with a short vector $\widehat{\mathbf{x}}$ and a $\underline{\pm 1 \text{ vector } \widehat{\mathbf{y}} \in \{\pm 1\}^m}$ such that $\mathbf{A} \cdot \widehat{\mathbf{x}} \ \underline{+\mathbf{B} \cdot \widehat{\mathbf{y}}} = \widehat{\mathbf{t}}$.
3. rOM-ISIS assumption says that $\mathcal{A}$ cannot output $\ell+1$ distinct vector tuples $\{(\mathbf{x}_j, \mathbf{y}_j, \mathbf{t}_j)\}_{j\in[\ell+1]}$ such that $\mathbf{A} \cdot \mathbf{x}_j \ \underline{+\mathbf{B} \cdot \mathbf{y}_j} = \mathbf{t}_j$, $\mathbf{t}_j \in T$, $\mathbf{x}_j$ is sufficiently short, $\underline{\mathbf{y}_j \text{ is a } \pm 1 \text{ vector}}$, *and* $\mathcal{A}$ made at most ℓ preimage queries.

Intuitively, the attacker now cannot truly select the preimage vector arbitrarily since the challenger randomizes the *actual* target vector as $(\mathbf{t} - \mathbf{B} \cdot \mathbf{y})$, where $\mathbf{y}$ is a random ± 1 vector. Since the attacker receives the vector $\widehat{\mathbf{y}}$ used for randomization, it is unclear whether we can reduce it to the standard ISIS assumption.[9] However, our preliminary cryptanalysis (cf. full version shows that it is more robust when compared with the OM-ISIS assumption. We believe that this new formulation could serve as a better lattice analogue of the one-more RSA assumption [BNPS03]. For example, we can also prove that a mild adaptation of the Agrawal et al. [AKSY22] two-round blind signature scheme is still secure under rOM-ISIS assumption, and now we no longer have set the parameters as carefully to avoid simple attacks as was done in [AKSY22]. This further illustrates the flexibility of our new assumption. Later, in Sect. 6, we describe the assumption in full detail and also provide some preliminary cryptanalysis.

Our Final NIBS Construction. With the above strengthening of the one-more ISIS assumption, we make some slight changes to our core design. Each signer additionally samples a random ± 1 vector $\mathbf{y}$, and computes the preimage for the syndrome for $(\mathsf{pk}_R - \mathbf{B} \cdot \mathbf{y})$, instead of just pk_R, i.e., it samples $\mathbf{z}$ such that $\mathbf{C} \cdot \mathbf{z} = \mathsf{pk}_R - \mathbf{B} \cdot \mathbf{y}$. The signer then explicitly sets $\mathbf{z}$ as the presignature and $\mathbf{y}$ as the nonce. Given this, the receiver creates a NIZK proof π stating that, given $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{w}, \delta$, there exist vectors $\mathbf{x}, \mathbf{y}$ and $\mathbf{z}$ such that the following relation holds:

$$\mathbf{C} \cdot \mathbf{z} + \mathbf{B} \cdot \mathbf{y} = \mathbf{A} \cdot \mathbf{x} + \mathsf{H}(\delta) \ \wedge \ \mathbf{w} = \mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_\perp \\ \mathbf{z}_\perp \end{bmatrix} \ \wedge$$

$$\mathbf{y} = \begin{bmatrix} y_1 \ y_2 \cdots \end{bmatrix}^\mathsf{T}, \ \forall i : y_i \in \{\pm 1\} \ \wedge \ \|\mathbf{x}\|, \ \|\mathbf{z}\| \text{ are short.}$$

We describe our NIBS construction in detail in Sect. 7.1. We prove one-more unforgeability of our NIBS protocol under the rOM-ISIS assumption. Here, we run into a slight technical issue. Namely, when proving one-more unforgeability, the reduction will need to extract the adversary's $\ell + 1$ forgeries from the NIZK proof, but rewinding $\ell + 1$ times results in an exponential (in ℓ) soundness loss. Fortunately, one can easily apply the so-called "encryption-to-the-sky" trick [AKSY22, BLNS23a] to get straight-line extraction. More concretely, we modify the proof system to also require a proof of encryption to the witness $(\mathbf{x}, \mathbf{y}, \mathbf{z})$. Therefore the precise relation that the receiver must prove also includes $\mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{pke.pk}, \mathbf{x}||\mathbf{y}||\mathbf{z}; r)$, where the ciphertext ct and the PKE public key $\mathsf{pke.pk}$ also form part of the instance, and the encryption randomness r is included in the witness. As done is prior works, this can be efficiently proved using a linear relation. Finally, the receiver sets $(\mathbf{w}, \delta)$ as its message and (π, ct) as the corresponding signature. To verify a signature, one simply runs the verification algorithm for the NIZK.

In the security proof, the reduction now retains access to the PKE secret key. This allows the reduction to extract $\mathbf{x}_i$, $\mathbf{y}_i$ and $\mathbf{z}_i$ for all of the adversary's $\ell + 1$ forgeries. If the hash function is modeled as a random oracle, the reduction can simply answer all hash queries for $H(\delta)$ from the challenge set T; if it further

[9] Interestingly, one can easily show by a simple application of the leftover hash lemma [HILL99, DRS04, DORS08], that if the attacker does not receive $\widehat{\mathbf{y}}$, then this is as hard as the standard ISIS assumption.

sets the public matrix $\mathbf{A}$ as a matrix-matrix product of the $\mathsf{rOM\text{-}ISIS}$ challenge matrix $\mathbf{C}$, it can break $\mathsf{rOM\text{-}ISIS}$. Observe that the matrix $\mathbf{A}$, statistically hides the receiver's secret $\mathbf{x}$ (by the leftover hash lemma). In the security proof, we combine this fact with the zero-knowledge property of the NIZK proof system, and the semantic security of the encryption scheme to prove receiver blindness.

Remark 2. Independently, Bootle et al. [BLNS23b] proposed the ISIS_f assumption. Under this assumption, the adversary is given access to a preimage oracle that outputs a randomly chosen $\widehat{y} \in D$ (D some domain) and a short vector $\widehat{\mathbf{x}} \in \mathbb{Z}_q^n$ such that for (public) matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ sampled from some distribution and (public) function $f : D \to \mathbb{Z}_q^n$, $\mathbf{A} \cdot \widehat{\mathbf{x}} = f(\widehat{y})$. The assumption then requires that it is hard for an adversary to output a value $y \in D$ and a short vector $\mathbf{x}$ ($\neq \widehat{\mathbf{x}}$) satisfying $\mathbf{A} \cdot \mathbf{x} = f(y)$. Importantly, this assumption is dependent on the choice of f. For instance, if f is the linear map $\mathbf{y} \mapsto \mathbf{B} \cdot \mathbf{y}$ for $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{y} \leftarrow\!\!\$\, \mathbb{Z}_q^m$, then the ISIS_f is trivially broken by linearly combining the query responses. [BLNS23b] also define an interactive version (that is reducible to the non-interactive ISIS_f) where the adversary is allowed to query for preimages under specific targets $\widehat{\mathbf{t}} \in \mathbb{Z}_q^l$ and the oracle outputs $(\widehat{y}, \widehat{\mathbf{x}})$ such that $\mathbf{A} \cdot \widehat{\mathbf{x}} = f(\widehat{y}) + \mathbf{C} \cdot \widehat{\mathbf{t}}$ for matrix $\mathbf{C} \in \mathbb{Z}_q^{n \times l}$ of the challenger's choice. Under this characterization, one might hope to abstractly view the rOM-ISIS assumption as an instantiation of interactive-ISIS_f where the function f linearly maps $\widehat{\mathbf{y}} \in \{\pm 1\}^m$ to $-\mathbf{B} \cdot \widehat{\mathbf{y}}$. However, if $\mathbf{C}$ is the $n \times n$ identity matrix as in rOM-ISIS, there is a non-negligible probability that outputs under this map can be efficiently linearly combined to give an *arbitrary* valid solution, so the ISIS_f problem is actually not hard for this function. On the other hand, the rOM-ISIS assumption remains hard as the adversary is also restricted to providing its (one-more) forgeries on a set T of target vectors chosen by the challenger.

Tagging Our NIBS Construction. We also discuss a simple modification to our NIBS construction to make it tagged. Recall that in tagged NIBS, the signer and receiver jointly agree on a value that will be treated as a public part of the signed message. To add such a public value τ to each blind signature, the signer computes a short preimage $\mathbf{z}$ such that $\mathbf{C} \cdot \mathbf{z} = \mathsf{pk}_R - \mathbf{B} \cdot \mathbf{y} + \mathsf{H}(\tau)$ using a secret trapdoor for $\mathbf{C}$. It then sends τ along with the preimage $\mathbf{z}$ and nonce $\mathbf{y}$ to the receiver. The receiver now includes τ in the instance of the NIZK relation, and generates an appropriate NIZK proof. The one-more unforgeability of this protocol follows by a similar reduction to $\mathsf{rOM\text{-}ISIS}$ (in the random oracle model). Of course, here we have the additional $\mathsf{H}(\tau)$ term, but notice that the one-more unforgeability reduction, both models the hash function to the adversary and chooses the tag τ. Thus the challenger can choose $\mathsf{H}(\tau)$ in a way that later allows it to extract a short preimage for its $\mathsf{rOM\text{-}ISIS}$ challenge using the (one-more) forgery. The argument for receiver blindness follows directly from receiver blindness of the NIBS counterpart.

2.6 Efficiency Comparisons for Our NIBS Schemes

As a proof-of-concept, we provide estimates for the various parameter sizes in our scheme. Just as [AKSY22], we instantiate our construction 7.1 with Falcon [FHK+17] for signatures, [LPS10] for Regev-style encryption and [LNP22b] for the NIZK for linear relations. In Table 2, we provide the public key, transcript and signature sizes for all our NIBS constructions. By weak blindness, we mean basic blindness [Han23], and by strong blindness, we mean our newly introduced definitions.

Table 2. Public key, transcript and signature sizes of our constructions.

Construction	$\lvert\mathsf{pk}_R\rvert$	$\lvert\mathsf{psig}\rvert+$ $\lvert\mathsf{nonce}\rvert$	$\lvert\sigma\rvert$	Blindness
Lattice-based (7.1)	1.6 KB	0.96 KB	68 KB	Weak
Circuit-private LHE (5.1)	$\mathsf{poly}(\lambda)$	$\mathsf{poly}(\lambda)$	~ 0.5 KB	Strong
General-purpose NIZKs (full version)	~ 2 KB	~ 0.5 KB	$\mathsf{poly}(\lambda)$	Strong

3 Preliminaries

We assume the reader is familiar with the standard cryptographic notions of PRFs, commitment schemes, PKE and digital signatures.

Notation. Let λ denote the security parameter, and PPT denote probabilistic polynomial-time. We denote the set of real numbers by $\mathbb{R}$ and the integers by $\mathbb{Z}$. We denote the set of all positive integers up to n as $[n] := \{1, \ldots, n\}$ and the set of all non-negative integers up to n as $[0, n] := \{0\} \cup [n]$.

For a vector $\mathbf{x}$ of even length, we write $\mathbf{x}_\top$ and $\mathbf{x}_\perp$ to be the top and bottom halves of $\mathbf{x}$ respectively, i.e. $\mathbf{x} = \begin{bmatrix}\mathbf{x}_\top \\ \mathbf{x}_\perp\end{bmatrix}$. Similarly, for any matrix $\mathbf{A} = [\mathbf{A}_\mathsf{L} \mid \mathbf{A}_\mathsf{R}] \in \mathbb{Z}_q^{n\times 2m}$, we denote its left and right halves as $\mathbf{A}_\mathsf{L}$ and $\mathbf{A}_\mathsf{R}$, respectively.

For a vector $\mathbf{x}$, we write its ℓ_2 norm as $\|\mathbf{x}\|_2$, often dropping the subscript and writing it simply as $\|\mathbf{x}\|$. We write the ℓ_∞ norm of $\mathbf{x}$ as $\|\mathbf{x}\|_\infty$.

3.1 Lattice Preliminaries

An m-dimensional lattice $\mathcal{L}$ is a discrete additive subgroup of $\mathbb{R}^m$. Given positive integers n, m, q and a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$, we let $\Lambda_q^\perp(\mathbf{A})$ denote the lattice $\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\cdot\mathbf{x} = \mathbf{0} \mod q\}$. For $\mathbf{u} \in \mathbb{Z}_q^n$, we let $\Lambda_q^\mathbf{u}(\mathbf{A})$ denote the coset $\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\cdot\mathbf{x} = \mathbf{u} \mod q\}$.

Discrete Gaussians. Let ς be any positive real number. The Gaussian distribution $\mathcal{D}_\varsigma$ with parameter ς is defined by the probability distribution function $\rho_\varsigma(\mathbf{x}) = \exp(-\pi\|\mathbf{x}\|^2/\varsigma^2)$. For any set $\mathcal{L} \subset \mathbb{R}^m$, define $\rho_\varsigma(\mathcal{L}) = \sum_{\mathbf{x}\in\mathcal{L}} \rho_\varsigma(\mathbf{x})$. The discrete Gaussian distribution $\mathcal{D}_{\mathcal{L},\varsigma}$ over $\mathcal{L}$ with parameter ς is defined by the probability distribution function $\rho_{\mathcal{L},\varsigma}(\mathbf{x}) = \rho_\varsigma(\mathbf{x})/\rho_\varsigma(\mathcal{L})$ for all $\mathbf{x} \in \mathcal{L}$.

The following lemma (Lemma 4.4 of [MR04, GPV08]) shows that if the parameter ς of a discrete Gaussian distribution is small, then any vector drawn from this distribution will be short (with high probability).

Lemma 1. *Let m, n, q be positive integers with $m > n$, $q \geq 2$. Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a matrix of dimensions $n \times m$, $\varsigma = \tilde{\Omega}(n)$ and $\mathcal{L} = \Lambda_q^{\perp}(\mathbf{A})$. Then*

$$\Pr[\|\mathbf{x}\| > \sqrt{m} \cdot \varsigma : \mathbf{x} \leftarrow\!\$\, \mathcal{D}_{\mathcal{L},\varsigma}] \leq \mathsf{negl}(n).$$

We will also require the following lemma (Lemma 4.4 of [Lyu12]) concerning the minimum-entropy of the discrete Gaussian distribution.

Lemma 2. *Let $\mathcal{D}_{\mathbb{Z}^m,\varsigma}$ be the discrete Gaussian distribution over $\mathbb{Z}^m$ for any $m > 1$, with variance ς. Then, for any $\varsigma \geq 3/\sqrt{2\pi}$ we have that $\mathbf{H}_\infty\left(\mathcal{D}_{\mathbb{Z}^m,\varsigma}\right) \geq m$.*

Lattice Trapdoors. Lattices with trapdoors are lattices that are statistically indistinguishable from randomly chosen lattices, but have certain 'trapdoors' that allow efficient solutions to hard lattice problems.

Definition 1 ([Ajt96, GPV08]). *For lattice parameters n, m, q with $m \geq \mathcal{O}(n \log q)$, a trapdoor lattice sampler consists of algorithms* TrapGen *and* SamplePre *with the following syntax:*

- $\mathsf{TrapGen}(1^n, 1^m, q) \rightarrow (\mathbf{A}, T_{\mathbf{A}})$*: The lattice generation algorithm is a randomized algorithm that takes as input the matrix dimensions n, m, modulus q, and outputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with a trapdoor $T_{\mathbf{A}}$.*
- $\mathsf{SamplePre}(\mathbf{A}, T_{\mathbf{A}}, \mathbf{u}, \varsigma) \rightarrow \mathbf{s}$*: The presampling algorithm takes as input a matrix $\mathbf{A}$, trapdoor $T_{\mathbf{A}}$, a vector $\mathbf{u} \in \mathbb{Z}_q^n$ and a parameter $\varsigma \in \mathbb{R}$ (which determines the length of the output vectors). It outputs a vector $\mathbf{s} \in \mathbb{Z}_q^m$ such that $\mathbf{A} \cdot \mathbf{s} = \mathbf{u}$ and $\|\mathbf{s}\| \leq \sqrt{m} \cdot \varsigma$.*

These algorithms must satisfy the following properties:

1. Well-Distributedness of Matrix. *The following distributions are statistically indistinguishable:*

$$\{\mathbf{A} : (\mathbf{A}, T_{\mathbf{A}}) \leftarrow\!\$\, \mathsf{TrapGen}(1^n, 1^m, q)\} \approx_s \{\mathbf{A} : \mathbf{A} \leftarrow\!\$\, \mathbb{Z}_q^{n \times m}\}.$$

2. Preimage Sampling: *For all $(\mathbf{A}, T_{\mathbf{A}}) \leftarrow\!\$\, \mathsf{TrapGen}(1^n, 1^m, q)$, if $\varsigma = \omega(\sqrt{n \cdot \log q \cdot \log m})$, then the following distributions are statistically indistinguishable:*

$$\{\mathbf{s} : \mathbf{u} \leftarrow\!\$\, \mathbb{Z}_q^n, \mathbf{s} \leftarrow\!\$\, \mathsf{SamplePre}(\mathbf{A}, T_{\mathbf{A}}, \mathbf{u}, \varsigma)\} \approx_s \mathcal{D}_{\mathbb{Z}^m,\varsigma}.$$

These properties are satisfied by the gadget-based trapdoor lattice sampler of [MP12] for parameters m such that $m = \Omega(n \cdot \log q)$.

Bonsai Lattice Trapdoors. In this work, we will rely on the bonsai trick for lattice trapdoors [ABB10a, CHKP10]. Briefly, using the standard Bonsai trick, one can sample preimage vectors with a special property that a portion of the preimage vector will be sampled as a random Gaussian vector with appropriate norm, rather than from a pre-defined lattice coset. Throughout the sequel, we will routinely sample matrices of dimensions $n \times 2m$ such as $\mathbf{A} \in \mathbb{Z}_q^{n\times 2m}$, where

$$(\mathbf{A}_{\mathsf{L}}, T_{\mathbf{A}_{\mathsf{L}}}) \leftarrow\!\$\; \mathsf{TrapGen}(1^n, 1^m, q), \quad \mathbf{A}_{\mathsf{R}} \leftarrow\!\$\; \mathbb{Z}_q^{n\times m}.$$

To create a preimage $\mathbf{s} \in \mathbb{Z}_q^{2m}$, for any vector $\mathbf{u} \in \mathbb{Z}_q^n$, such that $\mathbf{A} \cdot \mathbf{s} = \mathbf{u}$, we simply sample $\mathbf{s}_\perp \leftarrow\!\$\; \mathcal{D}_{\mathbb{Z}^m,\varsigma}$, and $\mathbf{s}_\top \leftarrow\!\$\; \mathsf{SamplePre}(\mathbf{A}_{\mathsf{L}}, T_{\mathbf{A}_{\mathsf{L}}}, \mathbf{u} - \mathbf{A}_{\mathsf{R}} \cdot \mathbf{s}_\perp, \varsigma)$.

We refer to the above bonsai-based lattice trapdoors as $\mathsf{bLT} = (\mathsf{bLT.TrapGen}, \mathsf{bLT.SamplePre})$, where the trapdoor generation and preimage sampling algorithms are defined as above. Clearly, the algorithms for bLT satisfy the standard well-distributedness property, as well as the preimage sampling property. Moreover, it satisfies the following stronger *half-preimage* well distribution property:

Half-Preimage Well Distributedness. For all $(\mathbf{A}, T_{\mathbf{A}}) \leftarrow\!\$\; \mathsf{bLT.TrapGen}(1^n, 1^{2m}, q)$, every vector $\mathbf{u} \in \mathbb{Z}_q^n$, every ς, the following distributions are identical:

$$\{\mathbf{s}_\perp : \mathbf{s} \leftarrow\!\$\; \mathsf{SamplePre}(\mathbf{A}, T_{\mathbf{A}}, \mathbf{u}, \varsigma)\} \equiv \mathcal{D}_{\mathbb{Z}^m,\varsigma}.$$

4 A Stronger Model for Non-Interactive Blind Signatures

In a NIBS system, a signer issues a random *presignature* psig for any receiver R with public key pk_R, such that the receiver R can extract a blind signature σ for a random message μ using its secret key sk_R. Syntactically, a non-interactive blind signature scheme consists of the following polynomial-time algorithms:

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. On input the security parameter λ, the global setup algorithm outputs a set of public parameters pp. All the remaining algorithms take pp as an input, but for notational clarity, we usually omit it as an explicit input.

$\mathsf{KeyGen}_S(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$. This corresponds to the signer's key generation algorithm. On input pp, it samples a public-secret key pair $(\mathsf{sk}, \mathsf{vk})$.

$\mathsf{KeyGen}_R(\mathsf{pp}) \to (\mathsf{sk}_R, \mathsf{pk}_R)$. This corresponds to the receiver's key generation algorithm. On input pp, it samples a public-secret key pair $(\mathsf{sk}_R, \mathsf{pk}_R)$.

$\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R) \to (\mathsf{psig}, \mathsf{nonce})$. This is a randomized algorithm that is run by the signer. It takes as input the signer's secret key sk as well as a receiver's public key pk_R. It then outputs a presignature psig along with nonce which represents (a portion of) the signer's random coins.

$\mathsf{Obtain}(\mathsf{sk}_R, \mathsf{vk}, (\mathsf{psig}, \mathsf{nonce})) \to (\mu, \sigma)$. This algorithm corresponds to the receiver's blind signature extraction algorithm. Given the receiver's secret key sk_R as an input, along with a verification key vk and presignature-nonce pair $(\mathsf{psig}, \mathsf{nonce})$, it outputs a message-signature pair (μ, σ) or aborts (in which case it outputs $\perp$).

$\mathsf{Verify}(\mathsf{vk}, \mu, \sigma) \to \{0, 1\}$. This is the signature scheme verification algorithm that takes as input a verification key and message-signature pair, and outputs 0/1.

Correctness. A non-interactive blind signature scheme satisfies correctness if for every security parameter $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \$ \, \mathsf{Setup}(1^\lambda)$, $(\mathsf{sk}, \mathsf{vk}) \leftarrow \$ \, \mathsf{KeyGen}_S(\mathsf{pp})$, $(\mathsf{sk}_R, \mathsf{pk}_R) \leftarrow \$ \, \mathsf{KeyGen}_R(\mathsf{pp})$, the following holds:

$$\Pr[\mathsf{Verify}(\mathsf{vk}, \mathsf{Obtain}(\mathsf{sk}_R, \mathsf{vk}, \mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R))) = 1] = 1,$$

where the probability is taken over the random coins of Issue and Obtain.

Remark 3 (Comparing with [Han23]*, and the reusability property).* The above formalization of non-interactive blind signatures is nearly identical to the syntax introduced by Hanzlik, except we do not regard nonce as an input supplied to the Issue algorithm but as an output. We essentially simplify the syntax by viewing nonce as a signer's *public (random) coins.* The Obtain algorithm receives both $(\mathsf{psig}, \mathsf{nonce})$ (as in [Han23]).

Paraphrasing [Han23], the purpose of nonce is to ensure reusability of a receiver's public key for obtaining multiple message-signature pairs from a single signer. We emphasize that both formulations are equivalent, and moreover, our formulation seems syntactically cleaner (and closer to syntax for traditional blind signatures) as well as helps defining an important reusability property that is necessary to avoid vacuous solutions. Further, we do not see any advantage in defining nonce as anything other than signer's public randomness since the goal here is to have the signing process be non-interactive, and treating nonce as an extra input is inconsistent with that goal.

Equivalence of both formalizations. It is straightforward to see that NIBS scheme satisfying the above syntax can be generically translated into satisfying the syntax from [Han23]. The idea is to generate the randomness for our Issue algorithm by evaluating a PRF on the nonce value and the public key pk_R provided as input in Hanzlik's version.[10] We describe this in detail in the full version.

A vacuous NIBS scheme. Consider a NIBS scheme where the Issue algorithm is deterministic and it always outputs the same nonce value (or, following Hanzlik's notation, the Issue algorithm ignores the nonce value entirely). Such a NIBS scheme clearly does not satisfy any meaningful notion of reusability, since for each receiver's public key a signer generates at most one presignature. Unfortunately, this is still a valid NIBS scheme as per existing definitions, and furthermore, it satisfies the nonce blindness property [Han23, Definition 17] vacuously, i.e. even if the adversary outputs $(\mathsf{psig}_0, \mathsf{nonce}_0) \neq (\mathsf{psig}_1, \mathsf{nonce}_1)$ it would still not be able to distinguish for (μ_b, σ_b). The main issue is that the existing definitions do not disallow schemes where the Issue and Obtain algorithms ignore the nonce parameter.

A simple and sound approach to capture reusability. Our proposal is to simply define reusability of NIBS schemes directly. Rather than making the nonce parameter explicit, we view it as a portion of the signer's random

[10] Concretely, $\mathsf{Issue}^{\mathsf{Hanzlik}}(\mathsf{sk}, \mathsf{pk}_R, \mathsf{nonce})$ outputs the presignature as $(\mathsf{psig}', \mathsf{nonce}')$ where $(\mathsf{psig}', \mathsf{nonce}') \leftarrow \mathsf{Issue}^{\mathsf{Ours}}(\mathsf{sk}, \mathsf{pk}_R; F_K(\mathsf{nonce}, \mathsf{pk}_R))$.

coins. Thus, we define reusability of a NIBS scheme as the property that any receiver can obtain two distinct messages (along with valid signatures) for two randomly generated presignature-nonce pairs (for the same receiver's public key) with all but negligible probability. Formally:

Definition 2 (Reusability). *A NIBS scheme $\mathcal{S}$ satisfies the reusability property, if there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, the following holds:*

$$\Pr\left[\begin{array}{l} \mathsf{nonce}_0 = \mathsf{nonce}_1 \\ \vee\ \mu_0 = \mu_1 \end{array} : \begin{array}{r} \mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda) \\ (\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{KeyGen}_S(\mathsf{pp}), (\mathsf{sk}_R, \mathsf{pk}_R) \leftarrow\!\!\$\ \mathsf{KeyGen}_R(\mathsf{pp}) \\ \forall b \in \{0,1\} : (\mathsf{psig}_b, \mathsf{nonce}_b) \leftarrow\!\!\$\ \mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R) \\ \forall b \in \{0,1\} : (\mu_b, \sigma_b) \leftarrow\!\!\$\ \mathsf{Obtain}(\mathsf{sk}_R, \mathsf{vk}, (\mathsf{psig}_b, \mathsf{nonce}_b)) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Next, we provide the standard notion of *one-more* unforgeability for blind signatures, specialized for the NIBS setting [Han23].[11]

Definition 3 (One-more unforgeability). *A NIBS scheme $\mathcal{S}$ satisfies one-more unforgeability, if for every* stateful admissible *PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, the following holds:*

$$\Pr\left[\begin{array}{l} \bigwedge_{i\in[\ell+1]} \mathsf{Verify}(\mathsf{vk}, \mu_i, \sigma_i) = 1 \\ \wedge\ \left(\bigwedge_{i\neq j\in[\ell+1]} \mu_i \neq \mu_j\right) \end{array} : \begin{array}{r} \mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda) \\ (\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\ \mathsf{KeyGen}_S(\mathsf{pp}) \\ \{(\mu_i, \sigma_i)\}_{i=1}^{\ell+1} \leftarrow\!\!\$\ \mathcal{A}^{O_{\mathsf{sk}}(\cdot)}(\mathsf{vk}) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $O_{\mathsf{sk}}(\cdot)$ takes as input a receiver's public key pk_{R_i}*, and outputs a presignature-nonce pair* $(\mathsf{psig}_i, \mathsf{nonce}_i)$ *by running* $\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_{R_i})$*, and $\mathcal{A}$ is an admissible adversary iff $\mathcal{A}$ makes at most ℓ queries to O_{sk}.*

In this work, we propose stronger notions of inter/intra-receiver blindness for NIBS schemes. The existing approaches to capture blindness for NIBS do not allow an adversary to learn any correlation between presignature-nonce pairs and their corresponding blind signature-message pairs. Unfortunately, if a server learns the receiver's identity for just one blind signature, then existing definitions are insufficient in providing any notion of blindness from such attacks. In order to protect from such advanced attackers that can bypass the blindness property for receivers on some selected blind signatures, we introduce stronger notions of inter/intra-receiver blindness properties that we refer to as *strong receiver/nonce blindness.*

Definition 4 (Strong receiver blindness). *A NIBS scheme $\mathcal{S}$ satisfies* strong *receiver blindness, if for every* stateful admissible *PPT adversary $\mathcal{A}$,*

[11] We want to remark that in [Han23], in the one-more unforgeability security experiment, the adversary can select nonce during each Issue query. In our definition, the challenger samples nonce since it is treated as randomness of Issue. However, both definitions are equivalent since a signer can use a PRF to generate the actual randomness from an input nonce as described in Remark 3.

there exists a negligible function $\mathsf{negl}(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, the following holds:*

$$\Pr\left[\begin{array}{l} \mathcal{A}^{O_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}(\cdot,\cdot,\cdot)}(\mu_{\hat{b}},\sigma_{\hat{b}},\mu_{1-\hat{b}},\sigma_{1-\hat{b}}) = \hat{b} : \\ \hfill \mathsf{pp} \leftarrow\$ \mathsf{Setup}(1^\lambda),\ \hat{b} \leftarrow\$ \{0,1\}, \\ \hfill \forall b \in \{0,1\} :\ (\mathsf{sk}_{R_b},\mathsf{pk}_{R_b}) \leftarrow\$ \mathsf{KeyGen}_R(\mathsf{pp}) \\ \hfill (\mathsf{vk},(\mathsf{psig}_b,\mathsf{nonce}_b)_b) \leftarrow\$ \mathcal{A}^{O_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}(\cdot,\cdot,\cdot)}(\mathsf{pk}_{R_0},\mathsf{pk}_{R_1}) \\ \hfill \forall b \in \{0,1\} :\ (\mu_b,\sigma_b) \leftarrow\$ \mathsf{Obtain}(\mathsf{sk}_{R_b},\mathsf{vk},(\mathsf{psig}_b,\mathsf{nonce}_b)) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where oracle $O_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}$*, on the i-th query* $(b^{(i)},\mathsf{vk}^{(i)},(\mathsf{psig}^{(i)},\mathsf{nonce}^{(i)}))$*, outputs* $\mathsf{Obtain}(\mathsf{sk}_{R_{b^{(i)}}},\mathsf{vk}^{(i)},(\mathsf{psig}^{(i)},\mathsf{nonce}^{(i)}))$*. That is,* $O_{\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}}$ *provides* $\mathcal{A}$ *oracle access to the* Obtain *algorithm w.r.t.* $\mathsf{sk}_{R_0},\mathsf{sk}_{R_1}$*. We say that* $\mathcal{A}$ *is an admissible adversary iff:*

- $\sigma_0,\sigma_1 \neq \perp$ *(i.e.,* Obtain *algorithm does not abort), and*
- $\mathsf{nonce}_0 \neq \mathsf{nonce}^{(i)}$ *and* $\mathsf{nonce}_1 \neq \mathsf{nonce}^{(i)}$ *for all i. (That is,* $\mathcal{A}$ *cannot make an* Obtain *query with nonce value to be either of the challenge nonce values.)*

Definition 5 (Strong nonce blindness). *A NIBS scheme* $\mathcal{S}$ *satisfies nonce blindness, if for every* stateful admissible *PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for every* $\lambda \in \mathbb{N}$*, the following holds:*

$$\Pr\left[\begin{array}{l} \mathcal{A}^{O_{\mathsf{sk}_R}(\cdot,\cdot)}(\mu_{\hat{b}},\sigma_{\hat{b}},\mu_{1-\hat{b}},\sigma_{1-\hat{b}}) = \hat{b} : \\ \hfill \mathsf{pp} \leftarrow\$ \mathsf{Setup}(1^\lambda),\ (\mathsf{sk}_R,\mathsf{pk}_R) \leftarrow\$ \mathsf{KeyGen}_R(\mathsf{pp}) \\ \hfill (\mathsf{vk},(\mathsf{psig}_b,\mathsf{nonce}_b)_b) \leftarrow\$ \mathcal{A}^{O_{\mathsf{sk}_R}(\cdot,\cdot)}(\mathsf{pk}_R),\ \hat{b} \leftarrow\$ \{0,1\} \\ \hfill \forall b \in \{0,1\} :\ (\mu_b,\sigma_b) \leftarrow\$ \mathsf{Obtain}(\mathsf{sk}_R,\mathsf{vk},(\mathsf{psig}_b,\mathsf{nonce}_b)) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where oracle O_{sk_R}*, on the i-th query* $(\mathsf{vk}^{(i)},(\mathsf{psig}^{(i)},\mathsf{nonce}^{(i)}))$*, outputs* $\mathsf{Obtain}(\mathsf{sk}_R,\mathsf{vk}^{(i)},(\mathsf{psig}^{(i)},\mathsf{nonce}^{(i)}))$*. That is,* O_{sk_R} *provides* $\mathcal{A}$ *oracle access to the* Obtain *algorithm w.r.t.* sk_R*. We say that* $\mathcal{A}$ *is an admissible adversary iff:*

- $\sigma_0,\sigma_1 \neq \perp$ *(i.e.,* Obtain *algorithm does not abort), and*
- $\mathsf{nonce}_0 \neq \mathsf{nonce}^{(i)}$ *and* $\mathsf{nonce}_1 \neq \mathsf{nonce}^{(i)}$ *for all i. (That is,* $\mathcal{A}$ *cannot make an* Obtain *query with nonce value to be either of the challenge nonce values.)*

Additional Definitions. The above unforgeability and blindness definitions are defined in the general chosen-key model where the attacker can specify arbitrary (possibly malformed) receiver and verification keys. In Appendix A we additionally consider definitions under the knowledge of secret keys (KOSK) model [MOR01,Bol03] for NIBS.

5 NIBS from Circuit Private LHE

We now describe our NIBS scheme from circuit-private LHE. The resulting construction has optimal-size signatures and strong security.

Tools Required. The construction relies on a pseudorandom function F, a signature scheme $\mathsf{S} = (\mathsf{S.Setup}, \mathsf{S.Sign}, \mathsf{S.Verify})$, a leveled homomorphic encryption scheme $\mathcal{LHE} = (\mathsf{LHE.Setup}, \mathsf{LHE.Enc}, \mathsf{LHE.Eval}, \mathsf{LHE.Dec})$ and a NIZKAoK proof system $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ for the following language:

Language $\mathcal{L}_1$

Instance: Each instance x is interpreted as an LHE evaluation key $\mathsf{lhe.ek}$, LHE ciphertexts ct and $\widehat{\mathsf{ct}}$ and randomness $r \in \{0,1\}^\lambda$.

Witness: Witness ω consists of a secret signing key sk and randomness ρ.

Membership: Let $C_{\mathsf{sk},r}$ encode the circuit $\mathsf{S.Sign}(\mathsf{sk}, F_{\cdot}(r))$ for a public pseudorandom function F. Then ω is a valid witness for x if the following is satisfied:

– $\widehat{\mathsf{ct}}$ is the homomorphic evaluation of the ciphertext ct over the circuit $C_{\mathsf{sk},r}$, ie., $\widehat{\mathsf{ct}} = \mathsf{LHE.Eval}(\mathsf{lhe.ek}, C_{\mathsf{sk},r}, \mathsf{ct}; \rho)$.

5.1 Construction

Below we describe our NIBS from circuit private LHE.

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$. It runs the setup algorithms for NIZK (for language $\mathcal{L}_1$):

$$\mathsf{nizk.crs} \leftarrow\!\!\$\, \mathsf{NIZK.Setup}(1^\lambda),$$

and outputs $\mathsf{pp} = \mathsf{nizk.crs}$.

$\mathsf{KeyGen}_S(\mathsf{pp}) \to (\mathsf{sk}, \mathsf{vk})$. The signer's setup algorithm runs the setup algorithm for the signature scheme S. Namely, it generates keys as $(\mathsf{sk}, \mathsf{vk}) \leftarrow\!\!\$\, \mathsf{S.Setup}(1^\lambda)$.

$\mathsf{KeyGen}_R(\mathsf{pp}) \to (\mathsf{sk}_R, \mathsf{pk}_R)$. The receiver's setup algorithm first samples a LHE key pair $(\mathsf{lhe.sk}, \mathsf{lhe.ek}) \leftarrow\!\!\$\, \mathsf{LHE.Setup}(1^\lambda, 1^d)$, and random PRF key $K \leftarrow\!\!\$\, \{0,1\}^\lambda$. (Here depth d is defined during the issue algorithm.) Next, it encrypts K as $\mathsf{ct} \leftarrow\!\!\$\, \mathsf{LHE.Enc}(\mathsf{lhe.sk}, K)$. Finally, it outputs receiver's secret key and public key as $\mathsf{sk}_R := (\mathsf{lhe.sk}, K)$ and $\mathsf{pk}_R := (\mathsf{lhe.ek}, \mathsf{ct})$.

$\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R) \to (\mathsf{psig}, \mathsf{nonce})$. The issue algorithm first samples a random message $r \leftarrow\!\!\$\, \{0,1\}^\lambda$. Let $C_{\mathsf{sk},r}(\cdot)$ be the following circuit (with key sk and message r hardwired)—$C_{\mathsf{sk},r}(K) \overset{\text{def}}{=} \mathsf{S.Sign}(\mathsf{sk}, F_K(r))$. That is, $C_{\mathsf{sk},r}$ runs the PRF function using the circuit input as the PRF key on message r, and then runs the signing algorithm to sign the output of the PRF. Let d denote the depth of the circuit $C_{\mathsf{sk},r}$. (This is the depth we set during the setup of the LHE scheme.)

The algorithm runs the homomorphic evaluation algorithm under uniformly random $\rho \leftarrow\$ \{0,1\}^{\lambda}$[12], and $(\mathsf{lhe.ek}, \mathsf{ct}) := \mathsf{pk}_R$, as follows:

$$\widehat{\mathsf{ct}} \leftarrow \mathsf{LHE.Eval}(\mathsf{lhe.ek}, C_{\mathsf{sk},r}, \mathsf{ct}; \rho),$$

and creates a NIZK proof π for the language $\mathcal{L}_1$ as

$$\pi \leftarrow\$ \mathsf{NIZK.Prove}(\mathsf{nizk.crs}, x = (\mathsf{pk}_R, \widehat{\mathsf{ct}}, r), \omega = (\mathsf{sk}, \rho))$$

Finally, it outputs the presignature $\mathsf{psig} := (\widehat{\mathsf{ct}}, \pi)$ and nonce $\mathsf{nonce} := r$.

$\mathsf{Obtain}(\mathsf{sk}_R, \mathsf{vk}, \mathsf{psig}, \mathsf{nonce}) \to (\mu, \sigma)$. The receiver parses $(\widehat{\mathsf{ct}}, \pi) := \mathsf{psig}$ and runs the NIZK verifier as $\mathsf{NIZK.Verify}(\mathsf{nizk.crs}, x = (\mathsf{pk}_R, \widehat{\mathsf{ct}}, \mathsf{nonce}), \pi)$ and aborts if it outputs 0. Otherwise it continues by computing the message as $\mu = F_K(r)$, where $(\mathsf{lhe.sk}, K) := \mathsf{sk}_R$ and $\mathsf{nonce} := r$. It then runs LHE decryption algorithm to compute the signature $\sigma \leftarrow\$ \mathsf{LHE.Dec}(\mathsf{lhe.sk}, \widehat{\mathsf{ct}})$, and it runs the signature verification algorithm to check that σ is a valid signature for μ under vk. That is, $\mathsf{S.Verify}(\mathsf{vk}, \mu, \sigma) = 1$. If the check fails, then it aborts. Otherwise, it outputs μ and σ as the corresponding message-signature pair.

$\mathsf{Verify}(\mathsf{vk}, \mu, \sigma) \to \{0,1\}$. The verification algorithm runs the signature scheme verifier and outputs whether $\mathsf{S.Verify}(\mathsf{vk}, \mu, \sigma) = 1$.

We now state the main theorem for this construction. The proof is given in the full version of this article.

Theorem 1 (Security).

1. *If* NIZK *satisfies zero knowledge, $\mathcal{LHE}$ is a circuit private LHE and* S *is a secure signature scheme, then Construction 5.1 is one-more unforgeable NIBS protocol in the KOSK model.*
2. *If* NIZK *is a NIZKAoK, $\mathcal{LHE}$ is a* IND-CPA *secure encryption scheme and F is a secure pseudorandom function, then Construction 5.1 is a strong receiver-blind NIBS protocol.*
3. *If* NIZK *is a NIZKAoK, $\mathcal{LHE}$ is a* IND-CPA *secure encryption scheme and F is a secure pseudorandom function, then Construction 5.1 is a strong nonce blind NIBS protocol.*

6 The Randomized One-More ISIS Assumption

In this work, we introduce a new variant of the OM-ISIS assumption with two goals—(i) protect from attackers that can ask for preimage queries on the same target vector more than once, and (ii) allow for re-randomization of the target vectors before answering the preimage query phase to make the assumption more robust. As discussed in [AKSY22, § 4.5], there are polynomial time attacks on the OM-ISIS assumptions for certain parameter regimes. At its core, all cryptanalysis efforts on OM-ISIS exploit the fact that the attacker can submit any

[12] This randomness is needed for circuit privacy.

target vector of its choice as a preimage query. Thus, an attacker can potentially request for preimage queries for short vectors, and use those to create an approximate trapdoor. However, the quality of trapdoor computed this way is much worse than the actual trapdoor, thus if the parameter β is set appropriately (i.e., sufficiently small), then an attacker cannot break the assumption since the approximate trapdoor will not give as short preimage vectors.

While existing cryptanalytic efforts do not succeed in breaking the assumption for the desired parameter regimes, we view the combinatorial and lattice-based attacks provided by [AKSY22] as partial evidence of existing assumption not being as robust. Moreover, we believe that while OM-ISIS is a very natural step towards defining lattice-analogue of the one-more-RSA assumption by Bellare et al. [BNPS03], there exists more robust instantiations for a family of one-more assumptions in lattice-based cryptography. To that end, we propose a strengthening of the OM-ISIS assumption that we call as randomized OM-ISIS (or, rOM-ISIS for short) assumption.

Our intuition is to draw more inspiration from GPV signatures. Recall that in GPV signatures, a signature is computed as a preimage of the output of a hash function (modeled as a random oracle). While modeling the hash function as a random oracle enables a reduction to plain ISIS by a standard RO programming argument, usage of any specific hash function (such as SHA-3) is not known to enable any efficient attacks. At a high level, the hash function protects the signer from simple algebraic manipulations of multiple preimage vectors to create a new valid preimage vector for a fresh hash output. A little more abstractly, this means that given preimage vectors $\{\mathbf{t}_i = \mathbf{A}_\varsigma^{-1}(H(\mu_i))\}_i$, it is unclear how to find short coefficients α_i s.t. $\sum \alpha_i \mathbf{t}_i = \mathbf{A}_\varsigma^{-1}(H(\mu^*))$. That is, the hash function prevents from creating a valid preimage to the hash function which is a short linear combination of other hash values.

Inspired by the intuitive structural guarantee provided by a hash function, we propose the following generalization of the OM-ISIS assumption.

Assumption 2 (randomized OM-ISIS)**.** *Let $q, n, m, \varsigma, \beta$ be functions of security parameter λ. Consider the following experiment:*

1. *The challenger uniformly samples two matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}$, and sends $\mathbf{A}, \mathbf{B}$ to adversary $\mathcal{A}$.*
2. *$\mathcal{A}$ adaptively makes queries of the following types to the challenger, in any order.*
   Syndrome queries. *$\mathcal{A}$ requests for a challenge vector, to which the challenger replies with a uniformly sampled vector $\mathbf{t} \leftarrow\$ \mathbb{Z}_q^n$. We denote the set of received vectors by S.*
   Preimage queries. *$\mathcal{A}$ queries a vector $\widehat{\mathbf{t}} \in \mathbb{Z}_q^n$, to which the challenger replies with a short vector $\widehat{\mathbf{x}} \in \mathbb{Z}_q^m$ and a ± 1 vector $\widehat{\mathbf{y}} \in \{\pm 1\}^m$ such that $\mathbf{A} \cdot \widehat{\mathbf{x}} + \mathbf{B} \cdot \widehat{\mathbf{y}} = \widehat{\mathbf{t}}$ and $\|\widehat{\mathbf{x}}\| \leq \varsigma\sqrt{m}$. Let ℓ denote the total number of preimage queries.*
3. *Finally, $\mathcal{A}$ outputs $\ell + 1$ tuples of the form $\{(\mathbf{x}_j, \mathbf{y}_j, \mathbf{t}_j)\}_{j \in [\ell+1]}$. $\mathcal{A}$ wins if*

$$\forall j \in [\ell+1], \qquad \mathbf{A} \cdot \mathbf{x}_j + \mathbf{B} \cdot \mathbf{y}_j = \mathbf{t}_j, \ \textit{and} \ \|\mathbf{x}_j\| \leq \beta, \mathbf{y}_j \in \{\pm 1\}^m \ \textit{and} \ \mathbf{t}_j \in S.$$

The $\mathsf{rOM\text{-}ISIS}_{q,n,m,\varsigma,\beta}$ *assumption states that for every PPT adversary* $\mathcal{A}$*, the probability that* $\mathcal{A}$ *wins is* $\mathsf{negl}(\lambda)$*.*

Intuitively, the new assumption says an attacker does not receive preimages for arbitrary target vectors $\widehat{\mathbf{t}}$ that it selects, but for publicly re-randomized vector $\mathbf{t}' = \widehat{\mathbf{t}} - \mathbf{B} \cdot \mathbf{y}$, where $\mathbf{y}$ is a random ± 1 vector (chosen by the challenger). We even provide the attacker with the vector $\mathbf{y}$. The point is that the ability to re-randomize the target vector, gives the challenger more flexibility in answering preimage queries. Now an attacker can win as long as it creates preimage vectors for any re-randomization of the random syndrome vectors. That is, we allow the attacker to also select any arbitrary ± 1 vector and use it to re-randomize each syndrome vector. The only constraint is that the vectors $\mathbf{y}_j$ are ± 1 vectors.

Unlike [AKSY22], we do not make any additional parameter restrictions. This is primarily due to the fact that we have been unable to find any practical attacks for any standard ISIS parameter regimes for the rOM-ISIS assumption. In our perspective, the condition that $\mathbf{y}_j$ must be ± 1 vectors prevents the algebraic attacks that were mounted on the OM-ISIS assumption. Additionally, one could view the $\mathbf{B} \cdot \mathbf{y}$ term as enforcing a structural property, on the target vectors, that a hash function enforces for GPV signatures.

In order to analyse the robustness of the randomized one-more ISIS assumption, we performed cryptanalysis for two categories of attacks presented against the one-more ISIS assumption by Agrawal et al. [AKSY22], and find that each of these attacks is in fact harder in the randomized one-more ISIS setting. We summarize our results below, and direct the reader to full version of this article for the complete analysis.

– For the lattice-based attack strategy, we are able to show a partial reduction to SIS (under certain conditions).
– For the combinatorial attack strategy, we give an **exponential** time algorithm for $\beta = \Theta\left(\sqrt{m/n} \cdot \varsigma\right)$. In contrast, the OM-ISIS assumption with has (i) a $\Omega(nq)$ time algorithm for $\beta = \Theta\left(\sqrt{m \cdot \left(1 + \frac{n \log q}{\log(Q/n^2)}\right)} \cdot \varsigma\right)$; and (ii) a $\Omega(n \lfloor \log q \rfloor)$ time algorithm for $\beta = \Theta(\sqrt{nm \log q} \cdot \varsigma)$.

7 Lattice-Based NIBS

We begin this section by describing our NIBS scheme that we prove secure under the $\mathsf{rOM\text{-}ISIS}$ assumption.

Let parameters $n, m, \varsigma, \beta = \varsigma\sqrt{m}$, a prime number q be functions of the security parameter λ such that the randomized one-more ISIS instance $\mathsf{rOM\text{-}ISIS}_{q,n,2m,\varsigma,3\sqrt{2}\beta}$ is hard. These parameters must satisfy the following constraints:

$$n = \mathsf{poly}(\lambda),\ m > n \log q + \lambda,\ \varsigma/m = \Omega\left(1\right),\ \beta < m\varsigma \tag{1}$$

Tools Required. Our construction relies on a public key encryption scheme $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$, a lattice trapdoor $\mathsf{bLT} = (\mathsf{TrapGen}, \mathsf{SamplePre})$, and a NIZK proof system $\mathsf{NIZK} = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ for the following language:

Language $\mathcal{L}_2$

Instance: Each instance x is interpreted matrices $\mathbf{C}, \mathbf{A}$ and $\mathbf{B}$, PKE public key $\mathsf{pke.pk}$, ciphertext ct, vector $\mathbf{w}$ and random string δ.
Witness: Witness ω consists of a secret vector $\mathbf{x}$, nonce vector $\mathbf{y}$, presignature vector $\mathbf{z}$ and randomness r.
Membership: ω is a valid witness for x if the following are satisfied:

- $\|\mathbf{z}\| \leq \sqrt{2}\beta$ and $\|\mathbf{x}\| \leq \sqrt{2}\beta/m$ and $\mathbf{y} \in \{\pm 1\}^{2m}$.
- ct is an encryption of $\mathbf{x}||\mathbf{y}||\mathbf{z}$ using randomness r, s.t. $\mathsf{ct} = \mathsf{PKE.Enc}(\mathsf{pke.pk}, \mathbf{x}||\mathbf{y}||\mathbf{z}; r)$
- $\mathbf{C} \cdot \mathbf{z} + \mathbf{B} \cdot \mathbf{y} = \mathbf{A} \cdot \mathbf{x} + \mathsf{H}(\delta)$ and $\mathbf{w} = \mathbf{A}_{\mathsf{L}} \cdot \mathbf{x}_{\perp} + \mathbf{A}_{\mathsf{R}} \cdot \mathbf{z}_{\perp}$

7.1 Construction

Below we describe our non-interactive blind signature scheme.

$\mathsf{Setup}(1^\lambda, n, m, q, \varsigma, \mathsf{H}) \rightarrow \mathsf{pp}$. It samples $\mathbf{A}, \mathbf{B} \leftarrow\$ \mathbb{Z}_q^{n \times 2m}$. Next, it runs the key generating algorithm of PKE and generates the public and secret key pair as

$$(\mathsf{pke.pk}, \mathsf{pke.sk}) \leftarrow\$ \mathsf{PKE.KeyGen}(1^\lambda).$$

Next, it generates $\mathsf{nizk.crs} \leftarrow\$ \mathsf{NIZK.Setup}(1^\lambda)$ and outputs:

$$\mathsf{pp} := (\mathbf{A}, \mathbf{B}, \mathsf{pke.pk}, \mathsf{nizk.crs}).$$

Note that H is the hash function which we model as a random oracle.

$\mathsf{KeyGen}_S(\mathsf{pp}) \rightarrow (\mathsf{sk}, \mathsf{vk})$. Runs the setup algorithm of lattice trapdoor and obtains

$$(\mathbf{T}_\mathbf{C}, \mathbf{C}) \leftarrow\$ \mathsf{bLT.TrapGen}(1^\lambda, n, 2m, q).$$

It outputs signer's secret key and verification key as $\mathsf{sk} := \mathbf{T}_\mathbf{C}$ and $\mathsf{vk} := \mathbf{C}$.

$\mathsf{KeyGen}_R(\mathsf{pp}) \rightarrow (\mathsf{sk}_R, \mathsf{pk}_R)$. It samples $\mathbf{x} \leftarrow\$ \mathcal{D}_{\mathbb{Z}_q^{2m}, \varsigma/m}$ and $\delta \leftarrow\$ \{0,1\}^\lambda$. Next, it computes

$$\mathbf{t} = \mathbf{A} \cdot \mathbf{x} + \mathsf{H}(\delta).$$

It outputs user's secret key and public key as $(\mathsf{sk}_R := (\mathbf{x}, \delta), \mathsf{pk}_R := \mathbf{t})$.

$\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R) \rightarrow (\mathsf{psig}, \mathsf{nonce})$. The issue algorithm samples a random ± 1 vector $\mathbf{y} \leftarrow\$ \{\pm 1\}^{2m}$. Next, using the signing key $\mathsf{sk} = \mathbf{T}_\mathbf{C}$ and the receiver's public key $\mathsf{pk}_R = \mathbf{t}$, it generates

$$\mathbf{z} \leftarrow \mathsf{bLT.SamplePre}(\mathbf{C}, \mathbf{T}_\mathbf{C}, \mathbf{t} - \mathbf{B} \cdot \mathbf{y}, \varsigma),$$

and outputs the presignature $\mathsf{psig} := \mathbf{z}$, and nonce as $\mathsf{nonce} := \mathbf{y}$.

$\mathsf{Obtain}(\mathsf{sk}_R, \mathsf{vk}, \mathsf{psig}, \mathsf{nonce}) \to (\mu, \sigma)$. It parses sk_R as $(\mathbf{x}, \delta)$. Then, assigns $\mathbf{C} := \mathsf{vk}$, $\mathbf{z} := \mathsf{psig}$, and $\mathbf{y} := \mathsf{nonce}$. It checks if $\mathbf{C} \cdot \mathbf{z} + \mathbf{B} \cdot \mathbf{y} = \mathbf{A} \cdot \mathbf{x} + \mathsf{H}(\delta)$, $\|\mathbf{z}\| \leq \sqrt{2}\beta$ and $\mathbf{y} \in \{\pm 1\}^{2m}$. If any check fails it aborts and outputs $\perp$. Otherwise, it generates

$$\mathsf{ct} \leftarrow \mathsf{PKE.Enc}(\mathsf{pke.pk}, \mathbf{x}||\mathbf{y}||\mathbf{z}; r)$$

from uniformly sampled randomness $r \leftarrow\$ \{0,1\}^{\lambda}$. The obtain algorithm sets $\mathbf{w}$ as $\mathbf{w} = \mathbf{A} \cdot \begin{bmatrix} \mathbf{x}_{\perp} \\ \mathbf{z}_{\perp} \end{bmatrix}$, where $\mathbf{x}_{\perp}, \mathbf{z}_{\perp} \in \mathbb{Z}_q^m$ and generates NIZK proof

$$\pi \leftarrow \mathsf{NIZK.Prove}\left(\mathsf{nizk.crs}, x := (\mathbf{C}, \mathbf{A}, \mathbf{B}, \mathsf{pke.pk}, \mathsf{ct}, \mathbf{w}, \delta), \omega := (\mathbf{x}, \mathbf{y}, \mathbf{z}, r)\right)$$

Finally, it outputs message $\mu := (\mathbf{w}, \delta)$ and signature $\sigma := (\pi, \mathsf{ct})$.

$\mathsf{Verify}(\mathsf{vk}, \mu, \sigma) \to \{0,1\}$. It parses μ as $(\mathbf{w}, \delta)$ and σ as (π, ct). The verification algorithm accepts and outputs 1 if and only if

$$\mathsf{NIZK.Verify}\left(\mathsf{nizk.crs}, x := (\mathbf{C}, \mathbf{A}, \mathbf{B}, \mathsf{pke.pk}, \mathsf{ct}, \mathbf{w}, \delta), \pi\right) = 1.$$

Otherwise, it outputs 0.

We now state the main theorem for this construction. The proof is given in the full version of this article.

Theorem 3 (Security).

1. *Assume that NIZK proof system* NIZK *satisfies soundness, lattice trapdoor* bLT *satisfies well-distributedness, and the* rOM-ISIS *assumption holds, then our Construction 7.1 is one-more unforgeable.*
2. *Assume that NIZK proof system* NIZK *satisfies zero knowledge property, and public key encryption scheme* PKE *is* IND-CPA *secure, then our construction 7.1 is receiver blind.*

Efficiency of Concrete Instantiation. We instantiated the protocol in Construction 7.1 using the same building blocks as [AKSY22]. Based on our analysis, we estimate the resulting signature size to be 67.7 KB. Please refer the full version for details.

Acknowledgements. We thank Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav for answering questions about the OM-ISIS assumption. We also thank Lucjan Hanzlik for helpful discussions on the motivation behind NIBS, potential applications enabled by varying levels of blindness security as well as pointing out a subtle issue in an earlier version of our NIBS definition.

Funding. Foteini Baldimtsi and Aayush Yadav were supported by NSF #2143287 and #2247304. Rishab Goyal was supported by OVCRGE at UW–Madison with funding from the Wisconsin Alumni Research Foundation.

A Knowledge of Secret Key Assumption

In the subsequent section, we will leverage the KOSK model [MOR01,Bol03] to prove the one-more unforgeability of our FHE-based Construction 5.1. Informally, this requires the attacker to specify the corresponding secret keys along with the respective verification/public keys. That is, the attacker supplies the receiver's secret key in unforgeability experiment and the signer's signing key in blindness experiments, respectively. We give the precise definitions next.

Definition 6 (Strong receiver blindness in KOSK). *Recall the strong receiver blindness security experiment from Definition 4. Consider another security experiment where $\mathcal{A}$ additionally provides a secret key $\mathsf{sk}^{(i)}$ along with its corresponding $\mathsf{vk}^{(i)}$ at the time of each oracle query. Moreover, $\mathcal{A}$ provides a secret key sk along with vk when declaring the challenge presignature-nonce pairs. We say $\mathcal{A}$ is admissible if $\mathsf{sk}^{(i)}$ (sk) is a valid secret key for $\mathsf{vk}^{(i)}$ (vk, respectively). Note that admissibility can be checked efficiently as the secret key $\mathsf{sk}^{(i)}$ can be regarded as the random coins of the setup algorithm.*

A NIBS scheme $\mathcal{S}$ satisfies strong *receiver blindness* in the KOSK model *if no admissible PPT adversary $\mathcal{A}$ wins in the above adapted security experiment with non-negligible probability.*

Definition 7 (Strong nonce blindness in KOSK). *A NIBS scheme $\mathcal{S}$ satisfies* strong *nonce blindness* in the KOSK model *if no admissible PPT adversary $\mathcal{A}$ wins in the adapted security experiment similar to in Definition 6 with non-negligible probability. Briefly, the adaptation is that the adversary provides a valid secret key associated with each verification key it outputs.*

Definition 8 (One-more unforgeability in KOSK). *A NIBS scheme $\mathcal{S}$ satisfies one-more unforgeability* in the KOSK model *if no admissible PPT adversary $\mathcal{A}$ wins in the adapted security experiment similar to in Definition 6 with non-negligible probability. Briefly, the adaptation is that the adversary provides a valid secret key associated with each receiver key it outputs.*

Our claim is that any NIBS scheme that is secure in the KOSK model can be generically upgraded to a fully secure scheme, i.e., one without the KOSK assumption using a NIZKAoK. We show this formally by building a secure NIBS scheme (without KOSK) given a NIBS scheme, $\mathsf{NIBS}' = (\mathsf{Setup}', \mathsf{KeyGen}'_S, \mathsf{KeyGen}'_R, \mathsf{Issue}', \mathsf{Obtain}', \mathsf{Verify}')$ that is secure in the KOSK model.

Language $\mathcal{L}'$

Instance: Each instance x is interpreted as a public key pk' and a bit $b \in \{0,1\}$.

Witness: Witness ω consists of a secret key sk', and randomness $\rho \in \{0,1\}^\lambda$.

Membership: Let C_0, C_1 be two circuits encoding $\mathsf{KeyGen}'_S(\mathsf{pp}', 1^\lambda; \cdot)$ and $\mathsf{KeyGen}'_R(\mathsf{pp}', 1^\lambda; \cdot)$ respectively. Then ω is a valid witness for x if the following is satisfied:

– $(\mathsf{sk}', \mathsf{pk}') = C_b(\rho)$

$\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp}$. The setup algorithm generates $\mathsf{pp}' \leftarrow\$ \mathsf{Setup}'(1^\lambda)$, and then runs the setup algorithms for NIZK (for language $\mathcal{L}'$) to generate $\mathsf{nizk.crs} \leftarrow\$ \mathsf{NIZK.Setup}(\mathsf{pp}', 1^\lambda)$ and outputs it as the public parameter pp of the protocol.

$\mathsf{KeyGen}_S(\mathsf{pp}) \rightarrow (\mathsf{sk}, \mathsf{vk})$. The signer's setup algorithm samples a random value $r_S \leftarrow\$ \{0,1\}^\lambda$ and runs $(\mathsf{sk}', \mathsf{pk}') \leftarrow \mathsf{KeyGen}'_S(\mathsf{pp}', 1^\lambda; r_S)$. It creates a NIZK proof $\pi_S \leftarrow\$ \mathsf{NIZK.Prove}(\mathsf{nizk.crs}, x := (\mathsf{pk}', 0), \omega := (\mathsf{sk}', r_S))$ for the language $\mathcal{L}'$. It outputs $\mathsf{sk} := \mathsf{sk}'$ and $\mathsf{vk} := (\mathsf{pk}', \pi_S)$.

$\mathsf{KeyGen}_R(\mathsf{pp}) \rightarrow (\mathsf{sk}_R, \mathsf{pk}_R)$. The receiver's setup algorithm samples a random value $r_R \leftarrow\$ \{0,1\}^\lambda$ and runs $(\mathsf{sk}', \mathsf{pk}') = \mathsf{KeyGen}'_R(\mathsf{pp}', 1^\lambda; r_R)$. It creates a NIZK proof $\pi_S \leftarrow\$ \mathsf{NIZK.Prove}(\mathsf{nizk.crs}, x := (\mathsf{pk}', 1), \omega := (\mathsf{sk}', r_R))$ for the language $\mathcal{L}'$. It outputs $\mathsf{sk} := \mathsf{sk}'$ and $\mathsf{vk} := (\mathsf{pk}', \pi_R)$.

$\mathsf{Issue}(\mathsf{sk}, \mathsf{pk}_R) \rightarrow (\mathsf{psig}, \mathsf{nonce})$. The issue algorithm parses the receiver's public key as $(\mathsf{pk}', \pi) := \mathsf{pk}_R$ and runs the NIZK verifier $\mathsf{NIZK.Verify}(\mathsf{nizk.crs}, x := (\mathsf{pk}', 1), \pi)$. If the verifier outputs 0, the signer aborts. Otherwise it continues to execute the issue algorithm.

$\mathsf{Obtain}(\mathsf{sk}_R, \mathsf{vk}, \mathsf{psig}, \mathsf{nonce}) \rightarrow (\mu, \sigma)$. The obtain algorithm parses the signer's public key as $(\mathsf{pk}', \pi) := \mathsf{vk}$ and runs the NIZK verifier $\mathsf{NIZK.Verify}(\mathsf{nizk.crs}, x := (\mathsf{pk}', 0), \pi_S)$. If the verifier outputs 0, the receiver aborts. Otherwise it continues to execute the obtain algorithm.

$\mathsf{Verify}(\mathsf{vk}, \mu, \sigma) \rightarrow \{0,1\}$. The verification algorithm runs $\mathsf{Verify}'(\mathsf{vk}, \mu, \sigma)$ and outputs whatever it outputs.

It is intuitively obvious that if NIBS' is a secure NIBS in the KOSK model and NIZK is an argument of knowledge, then the above construction $\mathsf{NIBS} = (\mathsf{Setup}, \mathsf{KeyGen}_S, \mathsf{KeyGen}_R, \mathsf{Issue}, \mathsf{Obtain}, \mathsf{Verify})$ is secure in the standard model. Essentially, in the security proof, instead of receiving the adversary's secret key (as in the KOSK model), the challenger must now run the NIZK extractor to obtain the adversary's secret. The rest of the proof would then proceed identically to that of NIBS'.

References

[ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

[ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 98–115, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

[AF96] Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 244–251, Kyongju, Korea, November 3–7, 1996. Springer, Heidelberg, Germany.

[AJL+12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 483–501, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[Ajt96] M. Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 99-108, New York, NY, USA, 1996. Association for Computing Machinery.

[AKSY22] Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 39–53, Los Angeles, CA, USA, November 7–11, 2022. ACM Press.

[AMPR14] Arash Afshar, Payman Mohassel, Benny Pinkas, and Ben Riva. Non-interactive secure computation based on cut-and-choose. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 387–404, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 132–145, Washington, DC, USA, October 25–29, 2004. ACM Press.

[BJOV18] Saikrishna Badrinarayanan, Abhishek Jain, Rafail Ostrovsky, and Ivan Visconti. Non-interactive secure computation from one-way functions. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part III*, volume 11274 of *LNCS*, pages 118–138, Brisbane, Queensland, Australia, December 2–6, 2018. Springer, Heidelberg, Germany.

[BL13] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 1087–1098, Berlin, Germany, November 4–8, 2013. ACM Press.

[BLNS23a] Ward Beullens, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Lattice-based blind signatures: Short, efficient, and round-optimal. Cryptology ePrint Archive, Report 2023/077, 2023. https://eprint.iacr.org/2023/077.

[BLNS23b] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A framework for practical anonymous credentials from lattices. In *CRYPTO 2023, Part II*, LNCS, pages 384–417, Santa Barbara, CA, USA, August 2023. Springer, Heidelberg, Germany.

[BNPS02] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme. In Paul F. Syverson, editor, *FC 2001*, volume 2339 of *LNCS*, pages 319–338, Grand Cayman, British West Indies, February 19–22, 2002. Springer, Heidelberg, Germany.

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46, Miami, FL, USA, January 6–8, 2003. Springer, Heidelberg, Germany.

[CDI+19] Melissa Chase, Yevgeniy Dodis, Yuval Ishai, Daniel Kraschewski, Tianren Liu, Rafail Ostrovsky, and Vinod Vaikuntanathan. Reusable non-interactive secure computation. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 462–488, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[CGT06] Sébastien Canard, Matthieu Gaud, and Jacques Traoré. Defeating malicious servers in a blind signatures based voting system. In Giovanni Di Crescenzo and Avi Rubin, editors, *FC 2006*, volume 4107 of *LNCS*, pages 148–153, Anguilla, British West Indies, February 27 – March 2, 2006. Springer, Heidelberg, Germany.

[Cha83] David Chaum. Blind signature system. In David Chaum, editor, *CRYPTO'83*, page 153, Santa Barbara, CA, USA, 1983. Plenum Press, New York, USA.

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, pages 523–552, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[DGS+18] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018.

[dK22] Rafaël del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 306–336, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 523–540, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

[FHK+17] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over ntru. Technical report, 2017.

[FHKS16] Georg Fuchsbauer, Christian Hanser, Chethan Kamath, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model from weaker assumptions. In Vassilis Zikas and Roberto De Prisco, editors, *SCN 16*, volume 9841 of *LNCS*, pages 391–408, Amalfi, Italy, August 31 – September 2, 2016. Springer, Heidelberg, Germany.

[FHS15] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 233–253, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[FHS19] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Heidelberg, Germany.

[Gha17] Essam Ghadafi. Efficient round-optimal blind signatures in the standard model. In Aggelos Kiayias, editor, *FC 2017*, volume 10322 of *LNCS*, pages 455–473, Sliema, Malta, April 3–7, 2017. Springer, Heidelberg, Germany.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206, Victoria, BC, Canada, May 17–20, 2008. ACM Press.

[GRS+11] Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 630–648, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[HAB+17] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. TumbleBit: An untrusted bitcoin-compatible anonymous payment hub. In *NDSS 2017*, San Diego, CA, USA, February 26 – March 1, 2017. The Internet Society.

[Han23] Lucjan Hanzlik. Non-interactive blind signatures for random messages. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 722–752, Lyon, France, April 23–27, 2023. Springer, Heidelberg, Germany.

[HBG16] Ethan Heilman, Foteini Baldimtsi, and Sharon Goldberg. Blindly signed contracts: Anonymous on-blockchain and off-blockchain bitcoin transactions. In Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, volume 9604 of *Lecture Notes in Computer Science*, pages 43–60. Springer, 2016.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HKKL07] Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 323–341, Amsterdam, The Netherlands, February 21–24, 2007. Springer, Heidelberg, Germany.

[HS14] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 491–511, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.

[HS21] Lucjan Hanzlik and Daniel Slamanig. With a little help from my friends: Constructing practical anonymous credentials. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2004–2023, Virtual Event, Republic of Korea, November 15–19, 2021. ACM Press.

[IKO+11] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 491–511, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.

[KNYY21] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Round-optimal blind signatures in the plain model from classical and quantum standard assumptions. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 404–434, Zagreb, Croatia, October 17–21, 2021. Springer, Heidelberg, Germany.

[Lin08] Yehuda Lindell. Lower bounds and impossibility results for concurrent self composition. *Journal of Cryptology*, 21(2):200–249, April 2008.

[LNP22a] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Efficient lattice-based blind signatures via gaussian one-time signatures. In Goichiro Hanaoka, Junji Shikata, and Yohei Watanabe, editors, *PKC 2022, Part II*, volume 13178 of *LNCS*, pages 498–527, Virtual Event, March 8–11, 2022. Springer, Heidelberg, Germany.

[LNP22b] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 71–101, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Heidelberg, Germany.

[LPS10] Vadim Lyubashevsky, Adriana Palacio, and Gil Segev. Public-key cryptographic primitives provably as secure as subset sum. In Daniele Micciancio, editor, *Theory of Cryptography*, pages 382–400, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Lyu12] Vadim Lyubashevsky. Lattice signatures without trapdoors. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 738–755, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[MOR01] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: Extended abstract. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 245–254, Philadelphia, PA, USA, November 5–8, 2001. ACM Press.

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718, Cambridge, UK, April 15–19, 2012. Springer, Heidelberg, Germany.

[MR04] D. Micciancio and O. Regev. Worst-case to average-case reductions based on gaussian measures. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 372–381, 2004.

[MR17] Payman Mohassel and Mike Rosulek. Non-interactive secure 2PC in the offline/online and batch settings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part III*, volume 10212 of *LNCS*, pages 425–455, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[MW16] Pratyay Mukherjee and Daniel Wichs. Two round multiparty computation via multi-key FHE. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 735–763, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[OPP14] Rafail Ostrovsky, Anat Paskin-Cherniavsky, and Beni Paskin-Cherniavsky. Maliciously circuit-private FHE. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 536–553, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[PS96] David Pointcheval and Jacques Stern. Provably secure blind signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 252–265, Kyongju, Korea, November 3–7, 1996. Springer, Heidelberg, Germany.

[PZ13] Christian Paquin and Greg Zaverucha. U-prove cryptographic specification v1.1 (revision 3). Technical report, Microsoft Corporation, December 2013.

[SC12] Jae Hong Seo and Jung Hee Cheon. Beyond the limitation of prime-order bilinear groups, and round optimal blind signatures. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 133–150, Taormina, Sicily, Italy, March 19–21, 2012. Springer, Heidelberg, Germany.

Dictators? Friends? Forgers.

Breaking and Fixing Unforgeability Definitions for Anamorphic Signature Schemes

Joseph Jaeger(✉) and Roy Stracovsky

School of Cybersecurity and Privacy, Georgia Institute of Technology, Atlanta, GA, USA
{josephjaeger,rstracovsky3}@gatech.edu

Abstract. Anamorphic signature schemes (KPPYZ, Crypto 2023) allow users to hide encrypted messages in signatures to allow covert communication in a hypothesized scenario where encryption is outlawed by a "dictator" but authentication is permitted. We enhance the security of anamorphic signatures by proposing two parallel notions of unforgeability which close gaps in existing security definitions. The first notion considers a *dictator* who wishes to forge anamorphic signatures. This notion patches a divide between the definition and a stated security goal of robustness (BGHMR, Eurocrypt 2024). We port two related BGHMR constructions to the signature scheme setting and show that one is secure when built from unpredictable signature schemes while the other is broken. The second notion considers a *recipient* who wishes to forge signatures. To motivate this notion, we identify a gap in an existing security definition from KPPYZ and present attacks that allow parties to be impersonated when using schemes erroneously deemed secure. We then formalize our new unforgeability definition to close this gap. Interestingly, while the new definition is only modestly different from the old one, the change introduces subtle technical challenges that arise when proving security. We overcome these challenges in our reanalysis of existing anamorphic signature schemes by showing they achieve our new notion when built from chosen-randomness secure signatures or with encryption that satisfies a novel ideal-model simulatability property.

1 Introduction

Cryptography provides a diverse set of tools and techniques that afford privacy, confidentiality, authenticity, and anonymity, among a myriad of other goals. Each goal can foster people's security, allowing them to focus on other needs and ultimately empowering them. This empowerment protects people from those who wish to violate their security, which can include governments. A controlling government may thus want to screen, monitor, or restrict the use of cryptography.

Persiano, Phan, and Yung [18] recently introduced a theoretical technique called anamorphic cryptography aimed at subverting or even dissuading such government control. Anamorphic cryptography envisions the dire setting of parties under the domain of a dictator who permits only limited forms of cryptography and can monitor all communication or force parties to give up their cryptographic keys in order to check for compliance. The authors then provide techniques for covert communication by hiding secret ciphertexts inside of other honestly generated ciphertexts. This work was followed up by [2,9,15,16,20] which refine and extend the original idea. Of note is [2] by Banfi, Gegier, Hirt, Maurer, and Rito (BGHMR), which proposes a notion of robustness (and many constructions achieving it) that allows parties using anamorphic cryptography to determine whether or not the outputs of a cryptosystem contain covert messages.

We focus on the notion of anamorphic signature schemes introduced by Kutylowski, Persiano, Phan, Yung, and Zawada (KPPYZ) [15]. They hypothesize a particularly restrictive setting in which the dictator bans encryption (or nullifies it by acting as a middlebox that decrypts and reencrypts all communication) but allows authentication, again with the caveat that they may coerce parties to surrender their signing keys. The authors show that covert communication is still possible, this time via anamorphic signature schemes which hide encrypted messages inside of innocuous signatures. Additionally, the authors provide schemes where the parties communicating covertly fully trust each other and schemes where that trust is eroded.

In this work, we improve the security of anamorphic signatures by proposing two parallel unforgeability definitions that comprehensively capture the anamorphic setting. The first notion considers unforgeability from the perspective of a dictator who knows each party's signing keys, while the second notion considers unforgeability from the perspective of an untrusted recipient privy to the hidden communication channel. As motivation for each notion, we first revisit two previously proposed security definitions from BGHMR and KPPYZ and observe crucial mismatches between the mathematical formalization of security and the expected deployment scenario. These mismatches allow natural (or even previously proposed) schemes that provably fulfill the old security definitions but would be trivially insecure in practice due to attacks we identify. Both of our new security definitions mend these gaps, and we show (with new definitions and proofs) conditions in which existing schemes satisfy our new security notions.

Provable Stealthiness. From an operational perspective, anamorphic cryptography is another theoretical tool in the basket of stealthy communication techniques. It is not meant to replace these techniques, but instead add another medium within which users can communicate covertly. This expands the stealthy bandwidth available to parties. From a security perspective, however, anamorphic cryptography provides strong provable security guarantees. One can consider (as Persiano, Phan, and Yung did in [18]) a simple stealthiness game in which a "dictator" (or simply any adversarial observer) is given access to either an honest channel or one with covert messages hidden inside, and they must identify which one they see. Anamorphic cryptosystems can easily achieve this definition because they (mis)use existing cryptographic primitives which themselves have provable security guarantees – anamorphic schemes can then derive provable stealthiness from properties that lead to these very guarantees. In contrast, analysis of traditional steganography per the definition above can rely on unrealistic or difficult-to-model assumptions about a channel distribution. For instance, a steganographic channel which takes images and replaces the least significant bits of pixel colors with pseudorandom encryptions is only provably secure under the tenuous assumption that the least significant bits are uniformly distributed (and that they even exhibit a distribution that can be easily modeled in the first place).

The way anamorphic cryptosystems naturally "inherit" provable stealthiness guarantees also allows us to study and construct anamorphic schemes that achieve *extended* provable security properties against broader attacks. For instance, as CCA notions of security extend CPA by considering active adversaries, we can explore stealthy channels in the presence of active censors or malicious recipients. This analysis has already been initiated with the nascent notions of robustness (discussed above) from BGHMR and private anamorphism (discussed later) from KPPYZ. It is these security notions that we critique and develop.

Anamorphic Signatures. An anamorphic signature scheme specifies (in addition to normal signature functionality) an anamorphic signing algorithm $\mathsf{asig} \leftarrow \mathsf{aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$ which uses a signing key sk to sign a benign message msg and "double key" dk to hide an anamorphic message amsg inside the anamorphic signature asig. A recipient knowing dk should be able to extract amsg from asig, while one not knowing dk should not be able to distinguish asig from a real signature *even if they know* sk. This stealthiness property is formalized via a real-or-anamorphic ($\mathsf{RoA\text{-}CAMA}$) game and knowledge of sk is motivated by viewing the attacker as a "dictator" who can compel disclosure of sk from citizens.

KPPYZ construct anamorphic signature schemes for many well-studied or deployed signature schemes. We view several of their constructions as instantiations of a transform we call RRep ("randomness replacement") which utilizes signatures from which one can extract the randomness used to sign. To sign anamorphically, amsg is encrypted with dk using a pseudorandom encryption scheme prE and the resulting ciphertext is used as the randomness for the signature scheme S's signing procedure. The recipient extracts randomness from signatures and decrypts. This complete procedure forms an anamorphic signature scheme denoted $\mathsf{aS} = \mathsf{RRep}[\mathsf{S}, \mathsf{prE}]$. Since prE produces pseudorandom ciphertexts, the anamorphic signatures will be indistinguishable from honest signatures. For some signature schemes the signing randomness is directly extractable from a signature, while in others the signing key is needed to extract the randomness. These schemes are labeled $\varnothing$-(randomness)-recoverable and *sk*-(randomness)-recoverable respectively. When RRep is applied to an *sk*-(randomness-)recoverable signature scheme, the recipient must know sk. Such an anamorphic scheme is called "symmetric" and can only be used in a setting where the recipient is trusted to not use knowledge of sk to maliciously forge signatures. For non-symmetric schemes (e.g., RRep applied to ∅-recoverable signature schemes), KPPYZ define a natural security goal termed private anamorphism (that should hold in addition to RoA-CAMA security), which asks that a malicious recipient, given dk and honest signatures but not sk, should not be able to forge new signatures of their own. KPPYZ prove that any anamorphic signature scheme in which dk is independent of the signing key is necessarily private anamorphic (assuming the starting signature scheme was unforgeable). This captures RRep applied to ∅-recoverable schemes as a special case.

1.1 Strengthening Robustness to Dictator Unforgeability

Robustness for Anamorphic Signatures. We begin by adapting BGHMR's robustness definition to the signature scheme setting. At a high level, robustness asks that one cannot anamorphically decrypt *honest* ciphertexts (or signatures). This property holds in addition to RoA-CAMA stealthiness (rather than instead of it). The benefits, as stated by BGHMR, are twofold. The first is practical.

> "It is intuitively desirable that a special symbol ⊥ is output indicating that the ciphertext (intentionally) contains no covert message ... [as the schemes are] potentially already being actively used for regular communication (and only occasionally required to transmit covert messages, from some point in time onward)." [2]

The second is for security.

> "The dictator could trick receivers into reveling that they are indeed in possession of a double key by sending them normally encrypted messages and observing whether they show any reaction." [2]

With robustness codified for anamorphic signature schemes, we then introduce the necessary formalism to allow two of BGHMR's proposed transforms to act on signature schemes. The first we call RIdP (randomness identification with PRF) and the second we call RIdPX (randomness identification with PRF and XOR). These transforms are both stealthy (RoA-CAMA secure) and robust.

A Gap in Robustness. We reflect on the security motivation for robustness in the austere dictator setting envisioned for anamorphic cryptography and find that the robustness definition excludes reasonable attacks a dictator may perform. Specifically, there is no reason for the dictator to restrict themselves to sending normally encrypted messages to observe a reaction. The dictator knows the secret cryptographic key and can compute ciphertexts or signatures using whatever randomness they want (rather than using fresh randomness like in the honest case). More problematically, the dictator can take *existing* ciphertexts or signatures *which may hide encrypted anamorphic messages* and modify them before sending them to the recipient. We introduce a strengthened security definition called $\underline{\text{d}}$ictator $\underline{\text{u}}$n$\underline{\text{f}}$orgeability under $\underline{\text{c}}$hosen $\underline{\text{a}}$namorphic $\underline{\text{s}}$ignature $\underline{\text{a}}$ttack to capture these scenarios. In the DUF-CASA game, the dictator is given the verification and signing keys and *separate* anamorphic signing and anamorphic decryption oracles which they can query however they want. The dictator's goal is to forge a new message-signature pair that contains an anamorphic message hidden within it. Table 1 summarizes BGHMR's robustness notion (denoted ROB-CMA) and our proposed dictator unforgeability security notion (DUF-CASA). From the table, it is evident that DUF-CASA comprehensively addresses the limitations in robustness.

Table 1. Existing security notions for anamorphic signatures compared to our proposed notions. Check marks (✓) denote inputs and oracle access given to adversaries. Starred check marks (✓*) denote that the adversary accesses one of two oracles in a distinguishing game. Daggered check marks ($✓^{\dagger}$) denote that the oracle only accepts arguments output by other oracles

		vk	sk	dk	O_{Sign}		O_{aSign}	O_{aDec}
Stealthiness (RoA-CAMA)	[15,18]	✓	✓		✓*	$\simeq$	✓*	
Robustness (ROB-CMA)	[2]	✓			✓			$✓^{\dagger}$
Dictator Unforgeability (DUF-CASA)	Our Work (Sect. 4)	✓	✓		N/A		✓	✓
Private Anamorphism (PA-CMA)	[15]	✓		✓	✓			N/A
Recipient Unforgeability (RUF-CAMA)	Our Work (Sect. 5)	✓		✓	✓		✓	N/A

Positive and Negative Results. We reanalyze the RIdP and RIdPX transforms ported from BGHMR with our proposed (DUF-CASA) dictator unforgeability notion and find that RIdP is secure provided the underlying signature scheme satisfies a security notion we term unpredictability under key compromise. In contrast, RIdPX is broken as a dictator can easily modify an anamorphic signature to contain a different (and potentially meaningful) anamorphic message! These results are summarized in Table 2. This separation indicates that our dictator unforgeability definition can parse out nuanced differences in schemes previously thought to have the same security.

1.2 Strengthening Private Anamorphism to Recipient Unforgeability

A Gap in Private Anamorphism. The second portion of our paper switches from the view of a dictator to the view of a malicious recipient. Our first contribution in this section shows the insufficiency of the KPPYZ notion of private anamorphism (discussed above as a property that may hold in addition to RoA-CAMA stealthiness). Recall that private anamorphism considers a recipient who is given dk and honest signatures and we desire that they cannot forge new signatures of their own. KPPYZ propose this definition so that the following holds.

"Anamorphic messages ... come with an implicit origin authentication as they are part of a signature that can only be produced by the owner of the signing key." [15]

There is a crucial gap between the definition of private anamorphism and this envisioned deployment scenario. In the former, the recipient *only receives real signatures* while over the course of the latter they also see anamorphic signatures. Motivated by this, we introduce a strengthened definition called recipient unforgeability under chosen anamorphic message attack. In the $\mathsf{RUF\text{-}CAMA}$ game, the attacker is given the double key and oracles for requesting *both* real and anamorphic signatures. Their goal is to produce forgeries. Table 1 summarizes KPPYZ's private anamorphism (formally denoted $\mathsf{PA\text{-}CMA}$) and our ($\mathsf{RUF\text{-}CAMA}$) recipient unforgeability security notions. From the table, we can see that $\mathsf{RUF\text{-}CAMA}$ patches an artificial limitation present in the definition of private anamorphism.

Breaking RRep. We start by asking whether ($\mathsf{RUF\text{-}CAMA}$) recipient unforgeability is different than private anamorphism. As a simple separating example, consider an anamorphic signature scheme that appends its signing key to an anamorphic message before encrypting it. Upon seeing a single anamorphic signature, any recipient trivially recovers the sender's signing key, even though this scheme is private anamorphic. This result, however, does not resolve whether recipient unforgeability is *meaningfully* different from private anamorphism for *realistic* schemes. To address this, we investigate whether randomness replacement RRep applied to $\varnothing$-recoverable schemes is always recipient unforgeable.

Damningly, we find this is not the case. Consider using stateful counter-mode encryption $\mathsf{prE_{cm}}$ in RRep (a natural choice of scheme which provides pseudorandom ciphertexts). Then an attacker, knowing dk and the current counter value, can compute the pseudorandom pad $\mathsf{prF}(\mathsf{k}, \mathsf{ctr})$ ahead of time and thus maliciously select an anamorphic message so that the resulting ciphertext $\mathsf{ct} = \mathsf{prF}(\mathsf{k}, \mathsf{ctr}) \oplus \mathsf{amsg}$ equals any string of its choice. Consequently, an anamorphic signature scheme $\mathsf{aS} = \mathsf{RRep}[\mathsf{S}, \mathsf{prE_{cm}}]$ using this encryption scheme is certainly insecure with any digital signature scheme S that is insecure against chosen randomness attacks. CRA security extends the typical unforgeability notion to allow the attacker to pick the randomness when requesting signatures.

Recovering Positive Results. Next, we ask whether positive results can be recovered for the recipient unforgeability of $\mathsf{aS} = \mathsf{RRep}[\mathsf{S}, \mathsf{prE}]$ with $\varnothing$-recoverable schemes by restricting the choice of signature S or encryption scheme prE. We provide two such results by requiring a stronger security property of either the signature scheme or the encryption scheme, while assuming nothing additional of the other scheme. The first is a natural complement to our counter-mode encryption attack; if the signature scheme S is CRA secure, then RRep is recipient unforgeable. As dk is independent of sk, all that it and the anamorphic signing can do is allow the attacker to influence the randomness used by the signing algorithm. We prove CRA security of RSA-PSS (considered in KPPYZ) here and Rabin signatures in the full version of this work.

This leaves unclear whether $\varnothing$-recoverable schemes like Boneh-Boyen (considered in KPPYZ), which are not known to be CRA secure, can be recipient unforgeable when used in RRep. For such schemes, we take the opposite approach and assume a strong ideal-model simulatability property of the encryption scheme prE. As an example, consider the encryption scheme where $\mathsf{ct} = (r, \mathsf{msg} \oplus H(\mathsf{k}, r))$. If H is modeled as a random oracle, then using this in RRep with any $\varnothing$-recoverable, unforgeable signature scheme S (which doesn't itself use H) will be recipient-unforgeable. The forging reduction adversary simulates H internally and can emulate anamorphic signatures using normal signatures by extracting ct from the normal signature and reprogramming H to be consistent. We make this ideal-model analysis modular by extracting this core reprogrammability property to define a new notion of $\underline{\text{sim}}$ulatability with random $\underline{\text{c}}$ipher$\underline{\text{t}}$exts ($\mathsf{SIM\text{-}\$CT}$) which will be achieved by appropriate idealizations of typical randomized encryption schemes like counter-mode encryption or cipher block chaining. Our positive results for RRep are included in Table 2.

Dictator and Recipient Unforgeable Schemes. It is sensible to construct anamorphic signature schemes that are *both* dictator unforgeable and recipient unforgeable. We show that randomness identification with PRF transform RIdP ported from BGHMR is secure when instantiated with $\varnothing$-recoverable schemes (previously only considered with RRep). This result is shown in Table 2. On the flip side, we touch on schemes that are forgeable by dictators *and* recipients, even though they satisfy robustness and private anamorphism! This serves as a final demonstration of the value of our definitions.

Table 2. The security of anamorphic signature scheme transforms (adapted from BGHMR and KPPYZ) under our proposed notions. The ✓ symbol indicates security while ✗ indicates insecurity. The ✓* symbol indicates we provide positive results as long as the underlying components satisfy additional security requirements

	RoA-CAMA	DUF-CASA	RUF-CAMA
$\mathsf{RIdP}[\mathsf{S}, \mathsf{prF}]$	✓	✓*	✓*
$\mathsf{RIdPX}[\mathsf{S}, \mathsf{prF}]$	✓	✗	–
$\mathsf{RRep}[\mathsf{S}, \mathsf{prE}]$	✓	✗	✓*

1.3 Related Work

Conceptually, anamorphic cryptography (broadly construed) is deeply related to a variety of research areas such as steganography and subliminal channels

[10,19,22] and kleptography/algorithmic substitution attacks [1,3,4,21]. These connections can be useful for inspiring positive results (e.g. RRep is reminiscent of the IV-replacement attack of [4]). In the other direction, negative results could also likely be ported over. For example, its possible that anamorphic signatures could be somewhat prevented if users are required to use signature schemes with unique signatures [4,11] or for which cryptographic reverse firewalls [17] are known. While techniques transfer between these areas, the change in high level perspective also inspires different low level technical insights (e.g. CRA-secure signatures and SIM-$\mathsf{\$CT}$-secure encryption).

1.4 Outline of This Paper

In Sect. 2, we introduce our notation and discuss cryptographic background relevant to our work. We then discuss anamorphic signature schemes in Sect. 3 and highlight the randomness replacement transform RRep. We begin in Sect. 4 by adapting and strengthening robustness to a dictator unforgeability notion and prove that a previously proposed scheme is secure while another is not. In Sect. 5, we provide two private anamorphic schemes which are insecure in practical settings, highlighting a gap in the existing definition. We supply two results that show that RRep is still secure in a new recipient unforgeability notion, provided updated security requirements on the underlying primitives.

2 Notation and Preliminaries

2.1 Pseudocode, Sets, and Tables

We leverage pseudocode formalism [6]. For an algorithm A, we write $y \leftarrow A^O(x)$ to denote running A with oracle access to O on input x and assigning the output to y. If an input set (or space) is specified for x, we assume only values from that set can be passed into A. When A is stateful, $A(x : \mathsf{st})$ denotes running A on input x and state st, where the state st is passed by reference and can be updated by A. When A is probabilistic, $A(x; r)$ denotes running A on input x and seed r. Omitting r indicates that the seed is sampled uniformly. We write PPT for probabilistic polynomial time.

Adversaries (e.g. $\mathcal{A}$, $\mathcal{B}$) are algorithms which may make up to $q(\lambda)$ oracle queries given a security parameter λ. We will write $q(\lambda)$ as q. Due to identically named oracles in some proofs, we notationally clarify them by writing $O^{\mathcal{A}}_{\mathsf{Func}}$ to denote the Func oracle that an adversary $\mathcal{A}$ accesses. Given a security notion SEC, we may consider modified notions where the adversary can access oracles for underlying primitives (e.g. in the random oracle model). We notate this as $\mathsf{SEC}(O_1, O_2, \dots)$. For a game $\mathcal{G}$, we let $\Pr[\mathcal{G}(\lambda) \Rightarrow 1]$ denote the probability $\mathcal{G}$ outputs 1 for a security parameter λ.

For a scheme C, parameters pp (which we discuss shortly), and associated class of objects X (e.g. keys, messages, or signatures), we call the set of possible values the X space (key space, message space, signature space) and denote these with double struck capital letters ($\mathsf{C}.\mathbb{K}_{\mathsf{pp}}$, $\mathsf{C}.\mathbb{M}_{\mathsf{pp}}$, $\mathsf{C}.\mathbb{S}_{\mathsf{pp}}$). For a stateful algorithm $\mathsf{C.Alg}(\ldots : \mathsf{st})$, we denote the set of possible states st with $\mathsf{C}.\mathbb{ST}_{\mathsf{pp}}^{\mathsf{Alg}}$. For a probabilistic algorithm $\mathsf{C.Alg}(\ldots; r)$, we denote the set of possible random seeds r with $\mathsf{C}.\mathbb{R}_{\mathsf{pp}}^{\mathsf{Alg}}$.

For a distribution χ over a set $\mathbb{S}$, we write $s \leftarrow\!\!\$\, \chi(\mathbb{S})$ to denote that s is sampled from χ or $s \leftarrow\!\!\$\, \mathbb{S}$ χ is uniform over $\mathbb{S}$.

2.2 Cryptographic Primitives and Preliminaries

We proceed with the understanding that the reader is familiar with digital signature schemes, pseudorandom encryption, and pseudorandom functions. For digital signature schemes we consider strong unforgeability under chosen message attack (SUF-CMA). Pseudorandom encryption refers to a (randomized or stateful) symmetric encryption scheme that has ciphertexts indistinguishable from random under chosen plaintext attack (IND\$-CPA). Lastly, for pseudorandom functions we consider pseudorandom function security (PRF). Full details are provided in the full version of this work.

Several of the schemes we discuss require public parameters. Due to interdependence between the parameter generation of schemes used together, we simplify notation by defining a PPT global parameter generation algorithm $\mathsf{PublicParamGen}(1^\lambda)$ that takes in a security parameter λ and outputs public parameters pp usable by all of the primitives and constructions in this work. One can think of pp capturing specifications like concrete groups and lengths of hash function inputs and outputs that are baked into global standards outside of any individual's control.

We analyze multiple anamorphic signature schemes arrived at by applying generic transformations to the ElGamal signature scheme [12] and RSA-PSS [5]. Their plain versions are specified below.

ElGamal Signature Scheme. For a security parameter λ, suppose that the public parameter generation algorithm $\mathsf{PublicParamGen}(1^\lambda)$ outputs pp containing a prime p where $\log(p) \in \Theta(\lambda)$, a generator g of $\mathbb{Z}_p^*$, and a hash function $H : \{0,1\}^* \times \mathbb{Z}_p^* \to \mathbb{Z}_p$. The ElGamal signature scheme ElG is as defined in Fig. 1.

RSA-PSS. Let $\mathbb{P}_{\lambda_\ell}$ denote the set of λ_ℓ-bit primes and $\mathsf{RSA.eGen}$ output RSA public exponents. Suppose $\mathsf{PublicParamGen}(1^\lambda)$ outputs pp containing $\lambda_\ell \in \Theta(\lambda)$, positive integers λ_0, λ_1 such that $\lambda_0 + \lambda_1 \leq \lambda - 1$, as well as hash functions $H : \{0,1\}^* \times \{0,1\}^{\lambda_0} \to \{0,1\}^{\lambda_1}$ and $G : \{0,1\}^{\lambda_1} \to \{0,1\}^{\lambda-\lambda_1-1}$. We consider G as separate G_1 and G_2 which output the first λ_0 and remaining $\lambda - \lambda_0 - \lambda_1 - 1$ bits of G's outputs respectively. Then RSA-PSS is as defined in Fig. 1.

3 Background on Anamorphic Signatures

3.1 Anamorphic Signature Schemes

We now review anamorphic signature schemes [15] (KPPYZ). At a high level, the objective of anamorphic signatures is to appear like regular signatures except that a user knowing a secret symmetric key, called a double key dk, can encrypt a covert message into a signature asig. This covert message, called the anamorphic message amsg, can be recovered from the asig using dk.

These anamorphic messages should be undetectable by a "dictator" who can see all signatures, force parties to hand over their secret signing keys, and ask parties sign messages of the dictator's choosing. To realize this, the signature scheme must still sign and verify like a standard signature scheme. Furthermore, the verification key, signing key, and signatures must be indistinguishable from honestly generated ones.

```
ElG.KeyGen(pp)
1 : $x \leftarrow\!\!\$\ \mathbb{Z}_{p-2} \setminus \{0\}$
2 : $y \leftarrow g^x \pmod p$
3 : $\mathsf{vk} \leftarrow (\mathsf{pp}, y)$
4 : $\mathsf{sk} \leftarrow (\mathsf{pp}, x)$
5 : return $(\mathsf{vk}, \mathsf{sk})$

ElG.Sign(sk, msg)
1 : $(\mathsf{pp}, x) \leftarrow \mathsf{sk}$
2 : $\kappa \leftarrow\!\!\$\ \mathbb{Z}^*_{p-1}$
3 : $r \leftarrow g^\kappa \pmod p$
4 : $\mathsf{ch} \leftarrow H(\mathsf{msg}, r)$
5 : $s \leftarrow (\mathsf{ch} - xr)\kappa^{-1} \pmod{p-1}$
6 : $\mathsf{sig} \leftarrow (r, s)$
7 : return sig

ElG.Verify(vk, msg, sig)
1 : $(\mathsf{pp}, y) \leftarrow \mathsf{vk}$
2 : $(r, s) \leftarrow \mathsf{sig}$
3 : $\mathsf{ch} \leftarrow H(\mathsf{msg}, r)$
4 : return $[\![g^{\mathsf{ch}} = y^r r^s]\!]$

RSA-PSS.KeyGen(pp)
1 : $p, q \leftarrow\!\!\$\ \mathbb{P}_{\lambda_\ell}$
2 : $N \leftarrow pq$
3 : $e \leftarrow \mathsf{RSA.eGen}(\mathsf{pp})$
4 : $d \leftarrow e^{-1} \pmod{\varphi(N)}$
5 : $\mathsf{vk} \leftarrow (\mathsf{pp}, N, e)$
6 : $\mathsf{sk} \leftarrow (\mathsf{pp}, N, d)$
7 : return $(\mathsf{vk}, \mathsf{sk})$

RSA-PSS.Sign(sk, msg)
1 : $(\mathsf{pp}, N, d) \leftarrow \mathsf{sk}$
2 : $r \leftarrow\!\!\$\ \{0,1\}^{\lambda_0}$
3 : $w \leftarrow H(\mathsf{msg}, r)$
4 : $(\alpha, \gamma) \leftarrow ((G_1(w) \oplus r), G_2(w))$
5 : $\mathsf{sig} \leftarrow (0 \| w \| \alpha \| \gamma)^d \pmod N$
6 : return sig

RSA-PSS.Verify(vk, msg, sig)
1 : $(\mathsf{pp}, N, e) \leftarrow \mathsf{vk}$
2 : $b \| w \| \alpha \| \gamma \leftarrow \mathsf{sig}^e \pmod N$
3 : $r \leftarrow G_1(w) \oplus \alpha$
4 : return $[\![b = 0]\!] \wedge [\![w = H(\mathsf{msg}, r)]\!]$
        $\wedge [\![\gamma = G_2(w)]\!]$
```

Fig. 1. ElGamal signature scheme ElG and RSA-PSS

We provide syntax and security definitions for stateless anamorphic signature schemes below (and for all subsequent definitions in this work) but note that some of our results consider stateful constructions. For these constructions, stateful syntax and security definitions are straightforward or we provide details if necessary. Full definitions and results for stateful anamorphic signatures can be found in the full version of this work.

Definition 1. *An* anamorphic signature scheme aS *is a signature scheme (specifying* $\mathsf{aS.KeyGen}$*,* $\mathsf{aS.Sign}$*, and* $\mathsf{aS.Verify}$*) with three additional PPT algorithms.*

- $\mathsf{aS.aKeyGen}(\mathsf{pp})$ *takes public parameters* pp *and generates the signing key* $\mathsf{sk} \in \mathsf{aS}.\mathbb{SK}_{\mathsf{pp}}$*, verification key* $\mathsf{vk} \in \mathsf{aS}.\mathbb{VK}_{\mathsf{pp}}$*, and double key* $\mathsf{dk} \in \mathsf{aS}.\mathbb{DK}_{\mathsf{pp}}$*. It then outputs* $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$*.*
- $\mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$ *takes the signing key* $\mathsf{sk} \in \mathsf{aS}.\mathbb{SK}_{\mathsf{pp}}$*, double key* $\mathsf{dk} \in \mathsf{aS}.\mathbb{DK}_{\mathsf{pp}}$*, message* $\mathsf{msg} \in \mathsf{aS}.\mathbb{M}_{\mathsf{pp}}$*, and anamorphic message* $\mathsf{amsg} \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$ *and outputs an anamorphic signature* $\mathsf{asig} \in \mathsf{aS}.\mathbb{AS}_{\mathsf{pp}}$*.*
- $\mathsf{aS.aDec}(\mathsf{dk}, \mathsf{msg}, \mathsf{asig})$ *takes the double key* $\mathsf{dk} \in \mathsf{aS}.\mathbb{DK}_{\mathsf{pp}}$*, message* $\mathsf{msg} \in \mathsf{aS}.\mathbb{M}_{\mathsf{pp}}$*, and anamorphic signature* $\mathsf{sig} \in \mathsf{aS}.\mathbb{AS}_{\mathsf{pp}}$ *and outputs an anamorphic message* $\mathsf{amsg} \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$*.*

In addition to standard signature scheme correctness, anamorphic schemes must satisfy the additional requirement that anamorphic messages are properly decryptable from anamorphic signatures.

Definition 2. *An anamorphic signature scheme* aS *is* correct *if standard signature scheme correctness holds and* $\mathsf{aS.aDec}(\mathsf{dk}, \mathsf{msg}, \mathsf{asig})$ *outputs* amsg *for all* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$*,* $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$*,* $\mathsf{msg} \in \mathsf{aS}.\mathbb{M}_{\mathsf{pp}}$*,* $\mathsf{amsg} \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$*, and* $\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$*.*

```
$\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$
1 : $b \leftarrow\!\!\$\ \{0,1\}$
2 : $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
3 : if $b = 0$ then
4 :     $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{aS.KeyGen}(\mathsf{pp})$
5 : else
6 :     $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
7 : $b^* \leftarrow \mathcal{A}^{O_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
8 : return $[\![b = b']\!]$
```

```
$O_{\mathsf{Sign}}(\mathsf{msg}, \mathsf{amsg})$
1 : if $b = 0$ then
2 :     $\mathsf{sig} \leftarrow \mathsf{aS.Sign}(\mathsf{sk}, \mathsf{msg})$
3 :     return $\mathsf{sig}$
4 : else
5 :     $\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$
6 :     return $\mathsf{asig}$
```

Fig. 2. Real-or-anamorphic game $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$ from KPPYZ

As mentioned prior, a dictator should not be able to tell whether signing or anamorphic signing is employed. This is formalized in KPPYZ via the real-or-anamorphic game $\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}$ defined in Fig. 2. The $\mathsf{RoA\text{-}CAMA}$ advantage for an anamorphic signature scheme aS is defined as

$$\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

Definition 3. *An anamorphic signature scheme* aS *is stealthy (in the* $\mathsf{RoA\text{-}CAMA}$ *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$ *is negligible.*

3.2 Constructing Anamorphic Signatures

KPPYZ introduces a diverse collection of ways to construct anamorphic signatures, including the Fiat-Shamir transform, rejection sampling, and the Naor-Yung paradigm. We view many of their proposed signature schemes as instantiations of a randomness replacement transform which constructs anamorphic signature schemes from a myriad of well-studied and deployed signature schemes.

Randomness replacement exploits the probabilistic nature of many signature schemes by encoding covert messages in the random coins used to generate signatures. We say signature schemes are randomness recoverable if they allow these random coins to be recovered by a recipient with sufficient knowledge and hence enable extraction of the covert message.

Randomness Recovery. While KPPYZ formulate randomness recovery in regard to Sigma protocols, we define it directly for signature schemes.

Definition 4. *A signature scheme* S *is* randomness recoverable *if it additionally specifies a PPT algorithm* $\mathsf{S.RRecov}$ *such that, for all* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$*,* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$*,* $\mathsf{msg} \in \mathsf{S}.\mathbb{M}_{\mathsf{pp}}$*,* $r \in \mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$*, and* $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$*,* r *can be recovered by computing* $r \leftarrow \mathsf{S.RRecov}(\mathsf{sig}, \mathsf{msg}, \mathsf{vk}, \mathsf{sk})$*.*

When $\mathsf{S.RRecov}$ requires sk to extract the randomness r, we label S as *sk*-(randomness)-recoverable. For some schemes, $\mathsf{S.RRecov}$ can recover the randomness without sk. In this case, we label S as $\varnothing$-(randomness)-recoverable and omit sk as an input to the algorithm.

Anamorphism via Randomness Replacement. The key idea for constructing anamorphic signature schemes via randomness replacement is to encrypt an anamorphic message into a pseudorandom ciphertext which is then used as the randomness when generating a signature. The randomness can then be recovered by a recipient through the randomness recovery function and decrypted to produce the anamorphic message. The formal construction is as follows.

Construction 1. *Consider the following needed to construct an anamorphic signature scheme via randomness replacement.*

- *Let* S *be a randomness recoverable signature scheme with randomness recovery function* $\mathsf{S.RRecov}$.

– *Let* prE *be a pseudorandom encryption scheme.*
– *Let* $\mathsf{PublicParamGen}$ *output parameters* pp *such that* $\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{C}_{\mathsf{pp}}$.

The anamorphic signature scheme $\mathsf{aS} = \mathsf{RRep}[\mathsf{S}, \mathsf{prE}]$ *is constructed as shown in Fig. 3, where the highlighted code is included if* S *is sk-recoverable.*

```
$\mathsf{aS.KeyGen} = \mathsf{S.KeyGen}$
$\mathsf{aS.Sign} = \mathsf{S.Sign}$
$\mathsf{aS.Verify} = \mathsf{S.Verify}$

$\mathsf{aS.aKeyGen(pp)}$
1 : $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen(pp)}$
2 : $\mathsf{k} \leftarrow \mathsf{prE.KeyGen(pp)}$
3 : $\mathsf{dk} \leftarrow (\mathsf{vk}, \mathsf{k}, \mathsf{sk})$
4 : return $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk})$

$\mathsf{aS}.\mathbb{DK}_{\mathsf{pp}} = \mathsf{S}.\mathbb{VK}_{\mathsf{pp}} \times \mathsf{prE}.\mathbb{K}_{\mathsf{pp}} \times \mathsf{S}.\mathbb{SK}_{\mathsf{pp}}$
$\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}} = \mathsf{prE}.\mathbb{M}_{\mathsf{pp}}$

$\mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk} = (\mathsf{vk}, \mathsf{k}, \mathsf{sk}), \mathsf{msg}, \mathsf{amsg})$
1 : $\mathsf{act} \leftarrow \mathsf{prE.Enc}(\mathsf{k}, \mathsf{amsg})$
2 : $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; \mathsf{act})$
3 : return $\mathsf{asig}$

$\mathsf{aS.aDec}(\mathsf{dk} = (\mathsf{vk}, \mathsf{k}, \mathsf{sk}), \mathsf{msg}, \mathsf{asig})$
1 : $\mathsf{act} \leftarrow \mathsf{S.RRecov}(\mathsf{asig}, \mathsf{msg}, \mathsf{vk}, \mathsf{sk})$
2 : $\mathsf{amsg} \leftarrow \mathsf{prE.Dec}(\mathsf{k}, \mathsf{act})$
3 : return $\mathsf{amsg}$
```

Fig. 3. Randomness replacement transform RRep

Note that Construction 1 requires that the given encryption scheme is pseudorandom with respect to the signature scheme's randomness space. This can be realized using encoding and rejection sampling techniques. The randomness replacement transform results in stealthy anamorphic signature schemes per Definition 3 (RoA-CAMA). Since we describe the construction slightly differently from KPPYZ, we accordingly provide adapted versions of Theorem 9 and 10 in KPPYZ (which prove stealthiness) in the full version of this work.

KPPYZ propose several anamorphic signature schemes that are instantiations of the randomness replacement transform. One symmetric anamorphic signature scheme is built on ElGamal. The scheme has a double key $\mathsf{dk} = (\mathsf{k}, x)$ where $\mathsf{sk} = x$ is the ElGamal signing key. Given an anamorphic signature (r, s) where $\kappa \leftarrow \mathsf{prE.Enc}(\mathsf{k}, \mathsf{amsg})$, $r \leftarrow g^{\kappa}$, and $s \leftarrow (H(\mathsf{msg}, r) - xr)\kappa^{-1}$, a trusted recipient who knows $\mathsf{sk} = x$ can extract $\kappa \leftarrow (H(\mathsf{msg}, r) - xr)s^{-1}$ and decrypt κ with k to obtain amsg. KPPYZ also propose a non-symmetric anamorphic signature scheme based on RSA-PSS. Given a signature sig, a recipient computes $b \| w \| \alpha \| \gamma \leftarrow \mathsf{sig}^{e} \pmod{N}$ and $r \leftarrow G_1(w) \oplus \alpha$. Thus parties can anamorphically communicate by replacing r with pseudorandom encryptions of amsg. No knowledge of the signing key sk is necessary to anamorphically decrypt.

Other Transforms. Anamorphic signature schemes can be constructed without RRep. For example, rejection sampling, touched on in KPPYZ, can generically construct a non-symmetric anamorphic signature scheme aS from *any* (sufficiently) probabilistic signature scheme. The core idea is to sample signatures sig until $\mathsf{prF}(\mathsf{k}, \mathsf{sig}) = \mathsf{amsg}$ for some pseudorandom function prF. This signature is returned as the anamorphic signature asig. Anamorphic decryption evaluates $\mathsf{amsg} \leftarrow \mathsf{prF}(\mathsf{k}, \mathsf{asig})$. Note that the size of the anamorphic message space $\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$ must be small for anamorphic signing to run in polynomial time.

In Sect. 4 we will adapt and discuss in detail two transforms from BGHMR to the signature scheme setting, neither of which can be expressed as instantiations of RRep. In contrast to rejection sampling, in which the sender "searches" for a signature that encrypts the anamorphic message they want to send, the transforms from BGHMR transfer this workload to the recipient who, given a signature, "searches" for the right anamorphic message encrypted within the signature.

4 Strengthening Robustness to Dictator Unforgeability

Following the introduction of anamorphic encryption in PPY [18], BGHMR [2] proposed a security notion dubbed robustness which allows parties communicating via anamorphic encryption to identify ciphertexts containing anamorphic messages. As discussed prior, BGHMR presents both practical and security considerations as motivation for robustness. A network with anamorphic channels presumably contains both honest and anamorphic ciphertexts, so an anamorphic party must be able to systematically distinguish such ciphertexts. Furthermore, a dictator may attempt to discern whether a party is using an anamorphic channel by sending them fresh ciphertexts, and hence a party should only decrypt anamorphic messages that were truly sent.

As robustness was only proposed for anamorphic *encryption* schemes, we begin by adapting and formalizing robustness and two constructions from BGHMR to the anamorphic signature scheme setting, before proposing a stronger notion and reanalyzing the two constructions.

4.1 Robustness (for Anamorphic Signatures)

The anamorphic signature scheme variant of robustness is captured by the $\mathcal{G}^{\mathsf{ROB\text{-}CMA}}$ game shown in Fig. 4. Following BGHMR, we present it as distinguishing game and consequently define the $\mathsf{ROB\text{-}CMA}$ advantage for an anamorphic signature scheme aS by

$$\mathbf{Adv}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

Definition 5. *An anamorphic signature scheme* aS *is robust (in the* $\mathsf{ROB\text{-}CMA}$ *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$ *is negligible.*

```
$\mathcal{G}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$
1 : $b \leftarrow\!\!\$ \{0,1\}$
2 : $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
3 : $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
4 : $b^* \leftarrow \mathcal{A}^{O_{\mathsf{Sign},\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk})$
5 : return $[\![b = b^*]\!]$

$O_{\mathsf{Sign},\mathsf{aDec}}(\mathsf{msg})$
1 : if $b = 0$ then return $\bot$
2 : $\mathsf{sig} \leftarrow \mathsf{aS.Sign}(\mathsf{sk}, \mathsf{msg})$
3 : $\mathsf{amsg} \leftarrow \mathsf{aS.aDec}(\mathsf{dk}, \mathsf{msg}, \mathsf{sig})$
4 : return $\mathsf{amsg}$
```

Fig. 4. Robustness game $\mathcal{G}^{\mathsf{ROB\text{-}CMA}}$ adapted from BGHMR

The BGHMR robustness notion considers an adversary not privy to secret keys that views *fresh, honest* ciphertexts that a party tries to anamorphically decrypt. The $\mathcal{G}^{\mathsf{ROB\text{-}CMA}}$ game captures this by giving the adversary the public verification key vk and access to an equivalent combined oracle $O_{\mathsf{Sign},\mathsf{aDec}}$ that signs new signatures and attempts to anamorphically decrypt them.[1] An adversary easily wins if any decryption is successful as it trivially sees that $b = 1$.

Neither the randomness replacement transform RRep nor rejection sampling achieve robustness as presented. While RRep can be modified to use an authenticated pseudorandom encryption scheme (though this may be difficult in the atomic setting due to size constraints), the nature of rejection sampling prohibits it from achieving robustness. This is because providing a signature as input to a pseudorandom function always results in some sort of output anamorphic message, and the inherent restriction on anamorphic message space size makes message authentication difficult.

4.2 Constructing Robust Anamorphic Signatures

BGHMR propose four transforms (and additional variants) to construct robust anamorphic encryption schemes. We describe one construction and variant and adapt them to signature schemes.

Randomness Identification with PRF. The first construction transforms any $\mathsf{IND\$\text{-}CPA}$-secure public-key encryption scheme PKE to a stateful robust anamorphic encryption scheme aPKE and roughly follows a decrypt-reencrypt paradigm. The double key dk consists of a pseudorandom function key k and public key pk. Users also maintain synchronized counters $\mathsf{ctr}_{\mathsf{Enc}}$ and $\mathsf{ctr}_{\mathsf{Dec}}$. To perform anamorphic encryption, the sender computes $r \leftarrow \mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{Enc}}, \mathsf{amsg}))$ and returns $\mathsf{act} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}, \mathsf{msg}, r)$. It also updates its counter $\mathsf{ctr}_{\mathsf{Enc}} \leftarrow \mathsf{ctr}_{\mathsf{Enc}}+1$. To perform anamorphic decryption of act, the receiver decrypts using sk to obtain msg and reencrypts msg using randomness $r' \leftarrow \mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{Dec}}, \mathsf{amsg}'))$ for all $\mathsf{amsg}' \in \mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}$. Once the receiver finds an amsg' that encrypts to

[1] In Table 1 we presented this oracle as separate O_{Sign} and O_{aDec} oracles where the latter only takes signatures output by the former. The formulation in Fig. 4 is equivalent.

act, it returns this amsg' or $\perp$ if no such amsg' exists, after which it updates $\mathsf{ctr}_{\mathsf{Dec}} \leftarrow \mathsf{ctr}_{\mathsf{Dec}} + 1$. Note that, similar to rejection sampling, the anamorphic message space $\mathsf{aPKE}.\mathbb{AM}_{\mathsf{pp}}$ must be sufficiently small for the scheme to run in polynomial time. Furthermore, both sender and receiver must ensure that their counters $\mathsf{ctr}_{\mathsf{Enc}}$ and $\mathsf{ctr}_{\mathsf{Dec}}$ are synchronized for correctness to hold.

In the anamorphic signature scheme setting, the recipient of an anamorphic signature may not have access to the sender's signing key (i.e. non-symmetric anamorphism) and hence cannot directly perform a resigning procedure analogous to reencryption. However, the transform above still works even if the recipient only *identifies* whether a given seed was used as encryption randomness (reencryption is simply a way to realize this). This technique, which we call randomness identification, is possible in many signature schemes.

Definition 6. *A signature scheme* S *is* randomness identifying *if it additionally specifies a PPT algorithm* $\mathsf{S.RIdtfy}$ *where* $\mathsf{S.RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig}, r')$ *outputs* 1 *if and only if* $r' = r$ *for all* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$*,* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$*,* $\mathsf{msg} \in \mathsf{S}.\mathbb{M}_{\mathsf{pp}}$*,* $r, r' \in \mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$*, and* $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$*.*

It is clear that $\varnothing$-recoverable signature schemes (where signing randomness is recoverable from sig without sk) are trivially randomness identifying. Of more interest is that ElGamal (and related signature schemes such as Schnorr) are also randomness identifying. Given a signature $\mathsf{sig} = (r, s) \leftarrow \mathsf{ElG.Sign}(\mathsf{sk}, \mathsf{msg}; \kappa)$, once can simply verify whether randomness κ' was used to generate sig by checking that $g^{\kappa'} = r$ and also ensuring that $\mathsf{ElG.Verify}(\mathsf{vk}, \mathsf{msg}, \mathsf{sig})$ outputs 1.

We can now construct robust anamorphic signature schemes by modifying the first transform from BGHMR, which we denote as randomness identification with PRF (RIdP). We also adapt a variant of the RIdP transform that doesn't input the anamorphic message into the pseudorandom function but instead XORs it. We denote this as randomness identification with PRF and XOR (RIdPX).

Construction 2. *Consider the following needed to construct an anamorphic signature scheme via randomness identification with PRF (and XOR).*

- *Let* S *be a randomness identifying signature scheme with randomness identification function* $\mathsf{S.RIdtfy}$*.*
- *Let* prF *be a pseudorandom function that takes in* $\boxed{\textit{3-tuple}}$ *or* $\boxed{\textit{2-tuple}}$ *messages where the first element of the tuple is an integer.*
- *Let* $\mathsf{PublicParamGen}$ *output parameters* pp *such that* $\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}} = \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}$ *and* $\mathsf{S}.\mathbb{M}_{\mathsf{pp}} = B$ *for* $\mathsf{prF}.\mathbb{M}_{\mathsf{pp}} = \boxed{\boxed{\mathbb{Z}_n \times B} \times C}$ *where* n *is a positive integer defined in* pp*.*

The anamorphic signature schemes $\boxed{\mathsf{aS} = \mathsf{RIdP}[\mathsf{S}, \mathsf{prF}]}$ *and* $\boxed{\mathsf{aS} = \mathsf{RIdPX}[\mathsf{S}, \mathsf{prF}]}$ *are constructed as shown in Fig. 5.*

Both the RIdP and RIdPX transforms achieve stealthiness (in the $\mathsf{RoA\text{-}CAMA}$ sense) and robustness. These results follow the proofs of security for the equivalent transforms in BGHMR, namely Lemma 4.1, 4.2, and 4.3. For completeness

$\mathsf{aS}.\mathbb{DK}_{\mathsf{pp}} = \mathsf{aS}.\mathbb{VK}_{\mathsf{pp}} \times \mathsf{prF}.\mathbb{K}_{\mathsf{pp}}$

$\boxed{\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}} = C}$ $\dbox{\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}} = \mathsf{prF}.\mathbb{R}_{\mathsf{pp}}}$

$\mathsf{aS}.\mathbb{ST}^{\mathsf{aSign}}_{\mathsf{pp}} = \mathsf{aS}.\mathbb{ST}^{\mathsf{aDec}}_{\mathsf{pp}} = \mathbb{Z}_n$

for $\mathsf{prF}.\mathbb{M}_{\mathsf{pp}} = \boxed{\dbox{\mathbb{Z}_n \times B} \times C}$

$\mathsf{aS.KeyGen} = \mathsf{S.KeyGen}$
$\mathsf{aS.Sign} = \mathsf{S.Sign}$
$\mathsf{aS.Verify} = \mathsf{S.Verify}$

```
aS.aKeyGen(pp)
1 : $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
2 : $\mathsf{k} \leftarrow \mathsf{prF.KeyGen}(\mathsf{pp})$
3 : $\mathsf{dk} \leftarrow (\mathsf{vk}, \mathsf{k})$
4 : $\mathsf{ctr}_{\mathsf{aSign}} \leftarrow 0$
5 : $\mathsf{ctr}_{\mathsf{aDec}} \leftarrow 0$
6 : return $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}, \mathsf{ctr}_{\mathsf{aSign}}, \mathsf{ctr}_{\mathsf{aDec}})$
```

```
aS.aSign(sk, dk = (vk, k), msg, amsg : ctr_aSign)
1 : $r \leftarrow \boxed{\mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{msg}, \mathsf{amsg}))}$
      $\dbox{\mathsf{amsg} \oplus \mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{aSign}}, \mathsf{msg}))}$
2 : $\mathsf{asig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$
3 : $\mathsf{ctr}_{\mathsf{aSign}} \leftarrow \mathsf{ctr}_{\mathsf{aSign}} + 1$
4 : return asig
```

```
aS.aDec(dk = (vk, k), msg, asig : ctr_aDec)
1 : $\mathsf{amsg} \leftarrow \perp$
2 : for $\mathsf{amsg}' \in \mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}$ do
3 :     $r' \leftarrow \boxed{\mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{msg}, \mathsf{amsg}'))}$
          $\dbox{\mathsf{amsg}' \oplus \mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{aDec}}, \mathsf{msg}))}$
4 :     if $\mathsf{S.RIdtfy}(\mathsf{vk}, \mathsf{msg}, \mathsf{asig}, r') = 1$ then
5 :         $\mathsf{amsg} \leftarrow \mathsf{amsg}'$
6 : $\mathsf{ctr}_{\mathsf{aDec}} \leftarrow \mathsf{ctr}_{\mathsf{aDec}} + 1$
7 : return amsg
```

Fig. 5. Randomness identification transforms $\boxed{\mathsf{RIdP}}$ and $\dbox{\mathsf{RIdPX}}$

(as our presentations of the transforms deal with a different underlying cryptographic object) we provide full ported proofs in the full version of this work.

4.3 Dictator Unforgeability

We now revisit the rationale behind robustness as discussed in BGHMR, specifically keeping in mind the dictator setting within which anamorphic cryptography aims to be secure. To reiterate, the first motivation is practical – many honest and anamorphic cryptographic outputs will be transmitted in a given network so parties should be able to systematically identify when a given ciphertext (or signature in our case) contains an anamorphic message encrypted within it. Robustness, combined with correctness, fully captures this goal. The second motivation is for security – a party should only be able to perform anamorphic decryption on ciphertexts (or signatures) which were truly anamorphically encrypted by another party, as a dictator might send ciphertexts or signatures to parties and see how they react. Robustness falls short of capturing this by excluding many practical actions a dictator may take.

In the public-key encryption setting considered in BGHMR, a dictator knows the public key and can encrypt messages themselves using randomness of their own choosing before sending them to users. The robustness game only computes fresh encryptions, hence precluding any attacks where a dictator selects particular randomness knowing that anamorphic parties may be inserting anamorphic messages into this randomness. This form of attack also applies to signature schemes as the anamorphic setting considers a dictator who obtains secret keys and can thus sign messages with randomness of their choosing.

Even more critically, it is unreasonable for a dictator to restrict themselves solely to encrypting new ciphertexts or signing new signatures. Instead, the dictator may take existing ciphertexts or signatures, which may contain hidden anamorphic messages that they have influenced, and maul them into other ciphertexts or signatures the dictator then uses for an attack. For instance, consider a situation where a dictator's actions have caused an anamorphic party to covertly send the message "Let's meet at 2:00 PM" via an anamorphic signature. The dictator who audits the signature may modify it so that the anamorphic message says "Let's meet at 4:00 PM" without even needing conclusive knowledge that the message is hidden inside.

To address these gaps between the robustness notion and desired security in the anamorphic setting, we introduce a new ciphertext-integrity-inspired security notion which we call dictator unforgeability under chosen anamorphic signature attack (DUF-CASA). In this notion a dictator gets access to an *anamorphic* signing oracle and may request anamorphic decryptions on any (real or anamorphic) signatures it obtains or generates itself with the goal of producing a *new* signature that doesn't anamorphically decrypt to $\perp$. This is formalized in the $\mathcal{G}^{\mathsf{DUF\text{-}CASA}}$ game in Fig. 6. Note that as the dictator now has access to an anamorphic signing oracle, they may try to return an unmodified anamorphic signature as their forgery.

```
$\mathcal{G}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$
1 : $S \leftarrow \varnothing$
2 : $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
3 : $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
4 : $(\mathsf{msg}^*, \mathsf{asig}^*) \leftarrow \mathcal{A}^{O_{\mathsf{aSign}}, O_{\mathsf{aDec}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
5 : $b_{\mathsf{valid}} \leftarrow [\![\mathsf{aS.aDec}(\mathsf{dk}, \mathsf{msg}^*, \mathsf{asig}^*) \neq \perp]\!]$
6 : $b_{\mathsf{new}} \leftarrow [\![(\mathsf{msg}^*, \mathsf{asig}^*) \notin S]\!]$
7 : return $b_{\mathsf{valid}} \wedge b_{\mathsf{new}}$
```

```
$O_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
1 : $\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$
2 : $S \leftarrow S \cup \{\mathsf{msg}, \mathsf{asig}\}$
3 : return $\mathsf{asig}$

$O_{\mathsf{aDec}}(\mathsf{msg}, \mathsf{asig})$
1 : $\mathsf{amsg} \leftarrow \mathsf{aS.aDec}(\mathsf{dk}, \mathsf{msg}, \mathsf{asig})$
2 : return $\mathsf{amsg}$
```

Fig. 6. Dictator unforgeability game $\mathcal{G}^{\mathsf{DUF\text{-}CASA}}$

Unlike robustness which is defined as an indistinguishability notion, we capture our security goal as an unforgeability game, thus we define the DUF-CASA advantage for an anamorphic signature scheme aS by

$$\mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 7. *An anamorphic signature scheme* aS *is dictator unforgeable (in the* $\mathsf{DUF\text{-}CASA}$ *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$ *is negligible.*

It is clear that dictator unforgeability for anamorphic signature schemes is a strictly stronger property than robustness. We formally capture this in the following theorem with full details in the full version of this work. The non-tight bound derives from the fact robustness allows the adversary to make many queries to the sign-then-anamorphic-decryption oracle while dictator unforgeability only allows the adversary to submit one forgery.

Theorem 1. *Let* aS *be an anamorphic signature scheme. Then for all PPT adversaries* $\mathcal{A}$ *that make* q_D *sign-then-anamorphic-decryption queries there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{ROB\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq q_D \mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS},\mathcal{B}}(\lambda).$$

4.4 RIdP with UP-KC-Secure Signatures

We now reexamine the security of both the randomness identification with PRF transform RIdP and randomness identification with PRF and XOR transform RIdPX (Fig. 5) in our comprehensive dictator unforgeability notion. We find that while the former is secure, the latter is not! This further motivates dictator unforgeability, showing that it can discriminate subtle security guarantees of previously proposed anamorphic schemes designed to have the same security.

In this section we focus on the dictator unforgeability of RIdP. To achieve this we require that the underlying signature scheme satisfy a property termed unpredictability under key compromise ($\mathsf{UP\text{-}KC}$), which asks that an adversary with access to a both the verification and secret signing keys of a signature scheme cannot predict a future signature on a message of their choice. This property is necessary for anamorphic signature schemes produced by the RIdP transform since its decryption procedure identifies amsg by looping through (effectively) fresh signatures and checking whether the input anamorphic signature equals the fresh signature computed in a given iteration of the loop. A dictator who (knowing sk) is able to predict a future signature can send this signature to the anamorphic decryption oracle. This would trigger the randomness identification check during the decryption loop, returning an anamorphic message and not $\perp$.

We introduce the formal notion of unpredictability under key compromise in Definition 8 which requires the $\mathcal{G}^{\mathsf{UP\text{-}KC}}$ game in in Fig. 7 which has the following $\mathsf{UP\text{-}KC}$ advantage defined by

$$\mathbf{Adv}^{\mathsf{UP\text{-}KC}}_{\mathsf{S},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{UP\text{-}KC}}_{\mathsf{S},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 8. *An signature scheme* S *is unpredictable under key compromise (in the* $\mathsf{UP\text{-}KC}$ *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{UP\text{-}KC}}_{\mathsf{S},\mathcal{A}}(\lambda)$ *is negligible.*

```
$\mathcal{G}^{\mathsf{UP\text{-}KC}}_{\mathsf{S},\mathcal{A}}(\lambda)$
1 : $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
2 : $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
3 : $(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{vk}, \mathsf{sk})$
4 : $\mathsf{sig}' \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}^*)$
5 : $\mathbf{return}\ [\![\mathsf{sig}^* = \mathsf{sig}']\!]$
```

Fig. 7. Unpredictability under key compromise game $\mathcal{G}^{\mathsf{UP\text{-}KC}}$

We are now ready to show that RIdP produces anamorphic signature schemes that are dictator unforgeable. We provide the theorem statement and a proof sketch below and defer a full proof to the full version of this work.

Theorem 2. *Let* S *be a randomness-identifying* $\mathsf{UP\text{-}KC}$*-secure signature scheme and* prF *be a pseudorandom function. In addition, let* $\mathsf{aS} = \mathsf{RIdP}[\mathsf{S}, \mathsf{prF}]$ *per Construction 2. Finally, let* $A(\lambda)$ *denote* $|\mathsf{aS}.\mathbb{AM}_{\mathsf{pp}}|$ *for* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$*. Then for all PPT adversaries* $\mathcal{A}$ *that make* q_D *anamorphic decryption queries there exist PPT adversaries* $\mathcal{B}_0$ *and* $\mathcal{B}_1$ *such that*

$$\mathbf{Adv}^{\mathsf{DUF\text{-}CASA}}_{\mathsf{aS}\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}_0}(\lambda) + (q_D + 1)A(\lambda)\mathbf{Adv}^{\mathsf{UP\text{-}KC}}_{\mathsf{S},\mathcal{B}_1}(\lambda)$$

where the $\mathcal{G}^{\mathsf{DUF\text{-}CASA}}$ *game is modified so that* $\mathcal{A}$*'s forgery is only not new if it was the* i^{th} *output of* O_{aSign} *and* $\mathcal{A}$ *has made* $i - 1$ *queries to* O_{aDec}.

Proof Sketch. The proof proceeds by a series of game hops from the $\mathcal{G}^{\mathsf{DUF\text{-}CASA}}$ game on aS to a final game that is bounded by the unpredictability advantage. The pseudorandomness of prF is used to transition to a game where signatures are seeded by a truly random function. Careful analysis shows that, if an adversary submits a query or forgery that would reuse an input to prF, a different check catches this before prF is called *or* the adversary loses the game by submitting a signature they have already seen as their forgery. Hence an adversary cannot distinguish between the $\mathcal{G}^{\mathsf{DUF\text{-}CASA}}$ game on aS and one where the signatures are all fresh. Finally, in this game with fresh signatures, an adversary who can submit a signature that will independently be computed during the anamorphic decryption loop can predict signatures under key compromise. Over the course of the game $(q_D + 1)A(\lambda)$ signatures are generated. □

Corollary 1. *Consider the ElGamal signature scheme* ElG *from Fig. 1 and let* prF *be an appropriate pseudorandom function. Then the anamorphic ElGamal signature scheme* $\mathsf{aElG} = \mathsf{RIdP}[\mathsf{ElG}, \mathsf{prF}]$ *is dictator unforgeable.*

4.5 An Attack on RIdPX

We have shown that RIdP, originally proposed to satisfy robustness in BGHMR, achieves the stronger dictator unforgeability notion (given a new security requirement) with ElGamal signatures as an example instantiation. We now show that

```
$\mathcal{G}^{\text{PA-CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$
1 : $S \leftarrow \varnothing$
2 : $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
3 : $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
4 : $(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{dk})$
5 : $b_{\mathsf{valid}} \leftarrow [\![\mathsf{aS.Verify}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1]\!]$
6 : $b_{\mathsf{new}} \leftarrow [\![(\mathsf{msg}^*, \mathsf{sig}^*) \notin S]\!]$
7 : return $b_{\mathsf{valid}} \wedge b_{\mathsf{new}}$

$O_{\mathsf{Sign}}(\mathsf{msg})$
1 : $\mathsf{sig} \leftarrow \mathsf{aS.Sign}(\mathsf{sk}, \mathsf{msg})$
2 : $S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$
3 : return $\mathsf{sig}$
```

Fig. 8. Private anamorphism game $\mathcal{G}^{\text{PA-CMA}}$ from KPPYZ

RIdPX, also proposed to satisfy robustness in BGHMR, does not achieve the dictator unforgeability notion when instantiated with ElGamal.

Let $\mathsf{aS}_{\mathsf{RIdPX\text{-}bad}} = \mathsf{RIdPX}[\mathsf{ElG}, \mathsf{prF}]$. An anamorphic signature of $\mathsf{aS}_{\mathsf{RIdPX\text{-}bad}}$ on a message msg and anamorphic message amsg is the signature $\mathsf{asig} = (r, s)$ with $r \leftarrow g^\kappa$ and $s \leftarrow (H(\mathsf{msg}) - xr)\kappa^{-1} \pmod{p-1}$ where $\kappa \leftarrow \mathsf{amsg} \oplus \mathsf{prF}(\mathsf{k}, (\mathsf{msg}, \mathsf{ctr}_{\mathsf{Enc}}))$. A dictator knows the signing key x and with this signature can extract $\kappa \leftarrow (H(\mathsf{msg}) - xr)s^{-1}$. They then compute $\kappa^* \leftarrow \kappa + \kappa' = (\mathsf{amsg} \oplus \kappa') \oplus \mathsf{prF}(k, (\mathsf{msg}, \mathsf{ctr}))$ and resign the same message msg with x to produce a signature asig^* which they submit as their forgery without querying the decryption oracle (allowing the counters $\mathsf{ctr}_{\mathsf{Enc}}$ and $\mathsf{ctr}_{\mathsf{Dec}}$ stay synchronized). Clearly asig^* decrypts to $\mathsf{amsg} \oplus \kappa'$ and not $\perp$! This is a complete and practical break of a reasonable instantiation of RIdPX and demonstrates the value of our proposed dictator unforgeability notion.

5 Strengthening Private Anamorphism to Recipient Unforgeability

We now turn to a different security notion for anamorphic signatures. This notion, called private anamorphism, was proposed in KPPYZ [15] and codifies a desired form of security for non-symmetric anamorphic schemes (ones where a recipient does not obtain sk as part of dk). Motivated through our analysis of this security definition, we will propose a new unforgeability definition – this time from the perspective of a recipient rather than a dictator. To begin our analysis, we identify a gap in the formal definition of private anamorphism and present an attack against a scheme that satisfies this definition.

5.1 Private Anamorphism

Private anamorphism aims to allow two parties to covertly communicate via anamorphic signatures without the need to give up their signing keys. Otherwise a malicious anamorphic party could conceivably forge signatures on messages of their choosing for any one of their correspondents, and in the case where the double key is shared between multiple users (which is mentioned in KPPYZ), they may insert their own anamorphic messages as well.

The definition of private anamorphism formulated in KPPYZ affords an even stronger security guarantee than simply preventing signing key recovery. It requires that a PPT adversary with a plain signing oracle cannot produce a forgery on any message whatsoever, even when given both the verification and double keys. KPPYZ's notion of private anamorphism, which we denote as $\mathsf{PA\text{-}CMA}$, is captured by the $\mathcal{G}^{\mathsf{PA\text{-}CMA}}$ game shown in Fig. 8. We define the $\mathsf{PA\text{-}CMA}$ advantage for an anamorphic signature scheme aS by

$$\mathbf{Adv}^{\mathsf{PA\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{PA\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 9. *An anamorphic signature scheme* aS *is private anamorphic (in the* $\mathsf{PA\text{-}CMA}$ *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{PA\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$ *is negligible.*

KPPYZ proved the following sufficiency result for private anamorphism.

Theorem 3 (Theorem 11 of KPPYZ [15]). *Let* aS *be an* $\mathsf{SUF\text{-}CMA}$*-secure anamorphic signature scheme where* $\mathsf{aS.aKeyGen}$ *is separable into the parallel and independent composition of algorithms that generate the signing keypair and double key. Then for all PPT adversaries* $\mathcal{A}$ *there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{PA\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{aS},\mathcal{B}}(\lambda).$$

In the following subsection, we present a simple attack against a scheme which satisfies Definition 9 and is hence private anamorphic before proposing a new definition called recipient unforgeability which better captures the abilities and knowledge of parties using anamorphic signatures. We then revisit the randomness replacement transform RRep which forms the basis of many anamorphic signature schemes proposed in KPPYZ and show that it is susceptible to a similar practical attack. We rectify this by providing new sufficiency results for anamorphic signatures arrived at via randomness replacement.

5.2 A Simple Attack

The attack we provide is contrived but nonetheless illustrates a gap between the definition of private anamorphism and reasonable abilities of an adversary. Suppose that Alice and Bob communicate using private anamorphic signatures to evade the watchful eye of a dictator, and Bob would like to steal Alice's signing key to impersonate her.

Alice and Bob construct their private anamorphic signature scheme $\mathsf{aS}_{\mathsf{Smpl\text{-}bad}}$ from an existing private anamorphic signature scheme aS. The two operate identically, except for $\mathsf{aS}_{\mathsf{Smpl\text{-}bad}}$'s anamorphic signing procedure which, upon input anamorphic message amsg, outputs $\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg} \| \mathsf{sk})$. Observe that $\mathsf{aS}_{\mathsf{Smpl\text{-}bad}}$ is stealthy (in the $\mathsf{RoA\text{-}CAMA}$-sense) as aS is itself stealthy and anamorphic signature schemes are stealthy regardless of what anamorphic messages are sent. Furthermore, $\mathsf{aS}_{\mathsf{Smpl\text{-}bad}}$ is private anamorphic because its plain signing oracle is identical to that of aS; we have only changed anamorphic signing. However, $\mathsf{aS}_{\mathsf{Smpl\text{-}bad}}$ is trivially broken in anamorphic settings. When Alice sends a single anamorphic signature asig to Bob, he is able to obtain her signing key sk by simply decrypting asig.

5.3 Recipient Unforgeability

The attack discussed above relies on the fact both real *and* anamorphic signatures are sent. Recall the goal of private anamorphism: protecting the signature scheme integrity of a sender Alice against a malicious recipient Bob. In the course of their covert communication, Alice will send Bob many anamorphic messages hidden within signatures. In fact, Bob may even influence the messages Alice sends (e.g. he might ask her to perform computation he may have rigged).

As a result, we argue that a new security notion, which we call recipient unforgeability under chosen anamorphic message attack (RUF-CAMA), is necessary to replace private anamorphism. The difference of RUF-CAMA from the existing PA-CMA notion of private anamorphism is that the adversary receives access to both a real signing and anamorphic signing oracle. Let aS be an anamorphic signature scheme, then the $\mathcal{G}^{\mathsf{RUF\text{-}CAMA}}$ is as shown in Fig. 9. The RUF-CAMA advantage for an anamorphic signature scheme aS is defined by

$$\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 10. *An anamorphic signature scheme* aS *is recipient unforgeable (in the* RUF-CAMA *sense) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$ *is negligible.*

```
$\mathcal{G}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda)$
1 : $S \leftarrow \varnothing$
2 : $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
3 : $(\mathsf{vk}, \mathsf{sk}, \mathsf{dk}) \leftarrow \mathsf{aS.aKeyGen}(\mathsf{pp})$
4 : $(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}}, O_{\mathsf{aSign}}}(\mathsf{pp}, \mathsf{vk}, \mathsf{dk})$
5 : $b_{\mathsf{valid}} \leftarrow [\![\mathsf{aS.Verify}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1]\!]$
6 : $b_{\mathsf{new}} \leftarrow [\![(\mathsf{msg}^*, \mathsf{sig}^*) \notin S]\!]$
7 : return $b_{\mathsf{valid}} \wedge b_{\mathsf{new}}$
```

```
$O_{\mathsf{Sign}}(\mathsf{msg})$
1 : $\mathsf{sig} \leftarrow \mathsf{aS.Sign}(\mathsf{sk}, \mathsf{msg})$
2 : $S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$
3 : return $\mathsf{sig}$

$O_{\mathsf{aSign}}(\mathsf{msg}, \mathsf{amsg})$
1 : $\mathsf{asig} \leftarrow \mathsf{aS.aSign}(\mathsf{sk}, \mathsf{dk}, \mathsf{msg}, \mathsf{amsg})$
2 : $S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{asig})\}$
3 : return $\mathsf{asig}$
```

Fig. 9. Recipient unforgeability game $\mathcal{G}^{\mathsf{RUF\text{-}CAMA}}$

It is clear that recipient unforgeability for anamorphic signature schemes is a strictly stronger property than private anamorphism, since the adversary gets access to an additional oracle in the former over the latter. We formally capture this in the following theorem.

Theorem 4. *Let* aS *be an anamorphic signature scheme. Then for all PPT adversaries* $\mathcal{A}$ *there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{PA\text{-}CMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{B}}(\lambda).$$

5.4 An Attack on RRep

Given the simple attack, we reexamine the randomness replacement transform RRep and find that it too is not necessarily secure in the envisioned deployment setting. Additionally, since the weakness relies on the transmission of a chosen anamorphic message, this further motivates our proposed new recipient unforgeability definition.

Like before, let Alice and Bob communicate using $\mathsf{PA\text{-}CMA}$-secure private anamorphic signatures, where Bob would like to steal Alice's signing key. This time Alice and Bob have chosen an anamorphic signature scheme arrived at via randomness replacement on a $\varnothing$-recoverable scheme (i.e. the signing randomness is replaced with pseudorandom encryptions of amsg, which Bob can extract and decrypt without sk). Alice believes this scheme protects her against malicious Bob as it is provably private anamorphic. However, the signature scheme contains a fatal flaw: when the randomness is equal to zero, the signing algorithm outputs its own secret key instead. Bob is then able to choose an anamorphic message that encrypts to the desired randomness and ask Alice to send it to him.

Stating the attack more concretely, let S be a $\varnothing$-recoverable signature scheme with signing key size equal to the size of signature randomness. S contains the weak point that signing with randomness equal to 0 outputs the secret key. Furthermore, let $\mathsf{prE_{cm}}$ be a pseudorandom encryption scheme built from a λ-bit pseudorandom function prF used in counter mode. That is, $\mathsf{prE_{cm}.Enc(k, msg : ctr_{Enc})}$ outputs $\mathsf{prF(k, ctr_{Enc}) \oplus msg}$ and sets $\mathsf{ctr_{Enc}} \leftarrow \mathsf{ctr_{Enc}}+1$ starting at $\mathsf{ctr_{Enc}} = 0$. We construct the anamorphic signature scheme $\mathsf{aS_{RRep\text{-}bad}} = \mathsf{RRep[S, prE_{cm}]}$ (with RRep defined in Fig. 3).

Straightforward reductions and and the separable and independent key generation framework from KPPYZ show that $\mathsf{aS_{RRep\text{-}bad}}$ is stealthy and private anamorphic. Details and full proofs can be found in the full version of this work.

Theorem 5. *For all PPT adversaries* $\mathcal{A}$ *that make* $q_S \leq 2^\lambda$ *signing queries there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{RoA\text{-}CAMA}}_{\mathsf{aS_{RRep\text{-}bad}},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{PRF}}_{\mathsf{prF},\mathcal{B}}(\lambda).$$

Theorem 6. *For all PPT adversaries* $\mathcal{A}$ *there exists a PPT algorithm* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{PA\text{-}CMA}}_{\mathsf{aS_{RRep\text{-}bad}},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{S},\mathcal{B}}(\lambda).$$

We now show that Bob can obtain Alice's signing key by asking her to send him a message of his choosing. Suppose that Alice has already sent ℓ messages to Bob. Bob first assigns $\mathsf{ctr}^*_{\mathsf{Enc}} \leftarrow \ell + 1$, then, knowing k, computes $\mathsf{amsg}^* \leftarrow \mathsf{prF}(\mathsf{k}, \mathsf{ctr}^*_{\mathsf{Enc}}) \oplus (0^\lambda)$. Observe that $\mathsf{prE}_{\mathsf{cm}}.\mathsf{Enc}(\mathsf{dk}, \mathsf{amsg}^*) = \mathsf{prF}(\mathsf{k}, \ell{+}1) \oplus \mathsf{prF}(\mathsf{k}, \ell{+}1) \oplus (0^\lambda) = 0$. Bob then asks Alice to send him amsg^*, triggering the weak point in the signature scheme allowing him to uncover her signing key.

Bob's chosen message attack involves two sources of weakness: he exploits the symmetric scheme by controlling the structure of the output and triggers an insecure point in the signature scheme. We discuss ways to mitigate either of these weaknesses in Sects. 5.5 and 5.6. More broadly, with a new recipient unforgeability definition, the sufficiency results of KPPYZ for private anamorphism no longer apply to desired settings. We show that the randomness replacement transform RRep is still secure against malicious recipients, provided updated assumptions for the underlying primitives.

5.5 RRep with SUF-CRA-Secure Signatures

The chosen message attack on RRep chiefly relies on the signature scheme's insecurity when randomness is chosen, as well as specific exploits of the pseudorandom encryption scheme by those who know the symmetric key. Here, we focus on the former by exploring signature schemes that are secure even when the signing randomness is chosen by an adversary.

```
$\mathcal{G}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{A}}(\lambda)$
1 : $S \leftarrow \varnothing$
2 : $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$
3 : $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
4 : $(\mathsf{msg}^*, \mathsf{sig}^*) \leftarrow \mathcal{A}^{O_{\mathsf{Sign}}}(\mathsf{pp}, \mathsf{vk})$
5 : $b_{\mathsf{valid}} \leftarrow [\![\mathsf{S.Verify}(\mathsf{vk}, \mathsf{msg}^*, \mathsf{sig}^*) = 1]\!]$
6 : $b_{\mathsf{new}} \leftarrow [\![(\mathsf{msg}^*, \mathsf{sig}^*) \notin S]\!]$
7 : return $b_{\mathsf{valid}} \wedge b_{\mathsf{new}}$
```

```
$O_{\mathsf{Sign}}(\mathsf{msg}, r)$
1 : $\mathsf{sig} \leftarrow \mathsf{S.Sign}(\mathsf{sk}, \mathsf{msg}; r)$
2 : $S \leftarrow S \cup \{(\mathsf{msg}, \mathsf{sig})\}$
3 : return $\mathsf{sig}$
```

Fig. 10. Existential forging under chosen randomness attack game $\mathcal{G}^{\mathsf{SUF\text{-}CRA}}$

Let S be a signature scheme signed using randomness space $\mathsf{S}.\mathbb{R}^{\mathsf{Sign}}_{\mathsf{pp}}$ for some parameters pp. Then the strong unforgeability under chosen randomness attack game $\mathcal{G}^{\mathsf{SUF\text{-}CRA}}$ is as defined in Fig. 10. The $\mathsf{SUF\text{-}CRA}$ advantage for a signature scheme S is defined as

$$\mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{A}}(\lambda) = \Pr[\mathcal{G}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{A}}(\lambda) \Rightarrow 1].$$

Definition 11. *A signature scheme* S *is* strongly unforgeable under chosen randomness attack *(or* $\mathsf{SUF\text{-}CRA}$-secure*) if, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{A}}(\lambda)$ *is negligible.*

We now show that randomness replacement can construct recipient unforgeable signatures from a large class of $\varnothing$-recoverable schemes with minimal assumptions needed for the pseudorandom encryption scheme. In particular, we prove that the anamorphic signature scheme formed from $\mathsf{RRep}[\mathsf{S},\mathsf{prE}]$ is private anamorphic if S is $\mathsf{SUF\text{-}CRA}$-secure and prE is pseudorandom. Full details are in the full version of this work.

Theorem 7. *Let* S *be an* $\mathsf{SUF\text{-}CRA}$*-secure* $\varnothing$*-recoverable signature scheme and* prE *be a pseudorandom encryption scheme. In addition, let* $\mathsf{aS} = \mathsf{RRep}[\mathsf{S},\mathsf{prE}]$ *as described in Construction 1. Then for all PPT adversaries* $\mathcal{A}$ *there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \le \mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{B}}(\lambda).$$

Proof. Given $\mathcal{A}$ against the $\mathsf{RUF\text{-}CAMA}$ security of aS we can construct an adversary $\mathcal{B}$ against the $\mathsf{SUF\text{-}CRA}$ security of S that simulates the $\mathsf{RUF\text{-}CAMA}$ game to $\mathcal{A}$ by computing all prE operations ($\mathsf{prE.KeyGen}$ and $\mathsf{prE.Enc}$) internally to compute the relevant r, which it then queries (along with the corresponding message) to its signing oracle. $\square$

Private Anamorphism from RSA-PSS. We show that the anamorphic variant of RSA-PSS (Fig. 1) derived via RRep is recipient unforgeable by showing that the $\mathsf{SUF\text{-}CRA}$ security of RSA-PSS is implied by the standard $\mathsf{SUF\text{-}CMA}$ security of RSA-PSS proved in [5]. We provide intuition for this below and a full proof in the full version of this work.

Lemma 1. *Let hash functions* H *and* G *that RSA-PSS uses be modeled as random oracles. Then for all PPT adversaries* $\mathcal{A}$ *that make up to* q_S *signing queries and* q_H *and* q_G *hash queries (to hash functions* H *and* G *respectively) there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{RSA\text{-}PSS},\mathcal{A}}(\lambda) \le (q_H+1)\left(\mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{RSA\text{-}PSS},\mathcal{B}}(\lambda) + q_T(q_T+q_G)(2^{-\lambda_0}+2^{-\lambda_1})\right)$$

for λ_0, λ_1 *in parameters* $\mathsf{pp} \leftarrow \mathsf{PublicParamGen}(1^\lambda)$ *and* $q_T = q_H + q_S$.

Proof Sketch. Given an $\mathsf{SUF\text{-}CRA}$ forger $\mathcal{A}$, we construct a PPT $\mathsf{SUF\text{-}CMA}$ forger $\mathcal{B}$ using a technique from the proof of full-domain hash signatures [5]. Since $\mathcal{A}$ is fully contained in $\mathcal{B}$, the latter can respond to $\mathcal{A}$'s queries how it chooses. However, from $\mathcal{A}$'s perspective it should be playing the $\mathsf{SUF\text{-}CRA}$ game, otherwise there is no guarantee it will output a correct forgery.

When $\mathcal{A}$ makes a signing query with randomness r, $\mathcal{B}$ is unable to make its own signing query for that particular randomness and instead makes a signing query to get a signature sig' on some randomness r'. The reduction $\mathcal{B}$ then simulates H and G to $\mathcal{A}$ so that they are consistent with sig' using randomness r. If $\mathcal{A}$ queries its random oracles on inputs not previously used during signing, $\mathcal{B}$ runs the signing procedure to create hash responses consistent with signatures.

Eventually, $\mathcal{A}$ will produce a forgery $(\mathsf{msg}^*, \mathsf{sig}^*)$. If $\mathcal{A}$ has non-negligible advantage then msg^* must have somehow been queried to the appropriate hash oracle previously, otherwise $\mathcal{A}$ would have no information on how to produce the forgery (since the hash is required for verification to work). Thus if $\mathcal{B}$ called its own signing oracle on every $\mathcal{A}$ hash query, then $\mathcal{A}$'s eventual forged signature could be on a signature $\mathcal{B}$ has received from its own signing oracle, even though $\mathcal{A}$ never saw this signature. That is, $\mathcal{B}$ loses its game if it forwards $\mathcal{A}$'s forgery, even if $\mathcal{A}$ itself wins. To rectify this, $\mathcal{B}$ selects one hash query to not respond to using a signing query, and hopes that $\mathcal{A}$ never queries the signing oracle on the corresponding message. □

Corollary 2. *Consider RSA-PSS from Fig. 1 and let* prE *be an appropriate pseudorandom encryption scheme. Then anamorphic RSA-PSS* $\mathsf{aRSA\text{-}PSS} = \mathsf{RRep}[\mathsf{RSA\text{-}PSS}, \mathsf{prE}]$ *is recipient unforgeable provided the RSA assumption holds.*

Tight proofs are known for RSA-PSS in the chosen message attack setting, but the $(q_H + 1)$ factor seems inherent when considering chosen randomness attacks for RSA-PSS. In the full version of this work we show that Rabin signatures are tightly chosen-randomness secure.

5.6 RRep with SUF-CMA-Secure Signatures

Again recall that the chosen anamorphic message attack on RRep (Sect. 5.4) leveraged both an insecurity in the underlying signature scheme and the pseudorandom encryption scheme. Having shown that signature schemes that satisfy a strong form of security can produce recipient unforgeable schemes via the RRep transform, we now show that standard $\varnothing$-recoverable signature schemes can still achieve private anamorphism with the right choice of pseudorandom encryption scheme.

Intuitively, Bob's ability to mount the chosen anamorphic message attack relied on his ability to choose anamorphic messages that encrypt to ciphertexts of his choice. Typical security definitions for symmetric encryption schemes cannot mitigate these attacks. For instance, indistinguishability from random bits does not capture this kind of attack because it only considers an adversary who does not have access to the symmetric key. Instead, we desire that the ciphertext distribution in some way appears independent of the symmetric key, even to those who know this key. In the standard model such a property seems ill-defined because of course the distribution must depend on the key by the fact that decryption must be correct. However, this kind of property is achievable in ideal models. Let us illustrate the core idea with an example.

We recall the example pseudorandom encryption scheme discussed in KPPYZ, which we refer to as r̲andomized h̲ash t̲hen X̲OR (RHtX). The scheme operates on λ-bit keys and messages and outputs 2λ-bit ciphertexts. Let H :

$\{0,1\}^{2\lambda} \rightarrow \{0,1\}^{\lambda}$ be a hash function. Then $\mathsf{RHtX.Enc}(\mathsf{k}, \mathsf{msg})$ outputs $\mathsf{ct} \leftarrow r \| \gamma$ where r is a random λ-bit string and $\gamma \leftarrow \mathsf{msg} \oplus H(\mathsf{k} \| r)$.

Let $\mathsf{aS} = \mathsf{RRep}[\mathsf{S}, \mathsf{RHtX}]$. Then we claim aS is recipient unforgeable if S is unforgeable (and $\varnothing$-recoverable), provided we model H as a random oracle which is not used by S. Given $\mathcal{A}$ against the recipient unforgeability of aS we can build a PPT $\mathcal{B}$ against the unforgeability of S. This $\mathcal{B}$ internally *simulates* H via *lazy sampling* and forwards non-anamorphic signing queries to its own signing oracle. Upon receiving an anamorphic signing query, $\mathcal{B}$ queries its own signing oracle to receive a signature, from which it extracts the randomness $\mathsf{act} = r \| \gamma$. It *programs* H so that $H(\mathsf{k} \| r) = \mathsf{msg} \oplus \gamma$ to ensure that RHtX decrypts properly. This reprogramming is undetectable unless $H(\mathsf{k} \| r)$ was already defined.

Note that RHtX is simply one fixed encryption scheme. For modularity, we wish to extract the core idea from above as a property to ask of (ideal model) encryption schemes. In practice, encryption schemes are built on block ciphers (possibly together with a hash function) which may be modeled as ideal ciphers, so we use a general syntax for an ideal primitive Prim (which we notate with syntax inspired by [13,14]). Prim specifies the following three PPT algorithms: $\mathsf{Prim.Init}$ initializes the primitive's state, $\mathsf{Prim.LS}$ defines lazy sampling, and $\mathsf{Prim.Prog}$ defines programming. In security games, honest scheme algorithms use the primitive's lazy sampling procedure $\mathsf{Prim.LS}$ while simulators and adversaries additionally use $\mathsf{Prim.Prog}$.

The <u>sim</u>ulatability with random <u>c</u>ipher<u>t</u>exts game $\mathcal{G}^{\mathsf{SIM\text{-}\$CT}}$ on a symmetric encryption scheme SE is as defined in Fig. 11. In this game the adversary must distinguish between a real world where the adversary receives honest encryptions and a simulated one where SE calls a simulation algorithm $\mathsf{SE.Sim}$ which attempts to program Prim to be consistent with randomly sampled ciphertexts. The adversary receives access to an SE encryption oracle and oracles $\mathsf{Prim.LS}$ and $\mathsf{Prim.Prog}$ for the underlying ideal primitive. Access to $\mathsf{Prim.Prog}$ is somewhat unnatural and is not essential to our results but is given for composability reasons identified by [8,13]. We notate the $\mathsf{SIM\text{-}\$CT}$ advantage for a symmetric encryption scheme SE built on ideal primitive Prim with

$$\mathbf{Adv}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{SE},\mathsf{Prim},\mathcal{A}}(\lambda) = 2\Pr[\mathcal{G}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{SE},\mathsf{Prim},\mathcal{A}}(\lambda) \Rightarrow 1] - 1.$$

Definition 12. *Let* SE *be a symmetric encryption scheme built on an ideal primitive* Prim*. It is* simulatable with random ciphertexts *(in the* $\mathsf{SIM\text{-}\$CT}$ *sense) if it specifies a PPT simulator* $\mathsf{SE.Sim}$ *such that, for all PPT adversaries* $\mathcal{A}$*, the function* $\mathbf{Adv}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{SE},\mathsf{Prim},\mathcal{A}}(\lambda)$ *is negligible.*

A full specification in our generalized notation and security proof of $\mathsf{SIM\text{-}\$CT}$ security for RHtX is given in the full version of this work. The key ideas in the proof extend to typical *randomized* modes of operation for block ciphers such as counter mode and cipher block chaining mode, though these modes exhibit ciphertext expansion due to the randomization which may be undesirable when trying to fit a ciphertext into signing randomness. Unfortunately, succinct symmetric encryption schemes are unrandomized, making them susceptible to the attack given in Sect. 5.4.

```
$\mathcal{G}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{SE},\mathsf{Prim},\mathcal{A}}(\lambda)$
1 : $b \leftarrow\$ \{0,1\}$
2 : $\mathsf{pp} \leftarrow\$ \mathsf{PublicParamGen}(1^\lambda)$
3 : $\mathsf{st} \leftarrow \mathsf{Prim.Init}(\mathsf{pp})$
4 : $\mathsf{k} \leftarrow \mathsf{SE.KeyGen}(\mathsf{pp})$
5 : $b^* \leftarrow \mathcal{A}^{O_{\mathsf{Enc}},O_{\mathsf{LS}},O_{\mathsf{Prog}}}(\mathsf{pp},\mathsf{k})$
6 : return $[\![b = b^*]\!]$
```

```
$O_{\mathsf{Enc}}(\mathsf{msg})$
1 : if $b = 0$ then
2 :     $\mathsf{ct} \leftarrow \mathsf{SE.Enc}^{O_{\mathsf{LS}}}(\mathsf{k},\mathsf{msg})$
3 : else
4 :     $\mathsf{ct} \leftarrow\$ \mathsf{SE}.\mathbb{C}_{\mathsf{pp}}$
5 :     $\mathsf{SE.Sim}^{O_{\mathsf{LS}},O_{\mathsf{Prog}}}(\mathsf{pp},\mathsf{k},\mathsf{msg},\mathsf{ct} : \mathsf{st})$
6 : return $\mathsf{ct}$
```

```
$O_{\mathsf{LS}}(x)$
1 : return $\mathsf{Prim.LS}(x : \mathsf{st})$
```

```
$O_{\mathsf{Prog}}(z)$
1 : return $\mathsf{Prim.Prog}(z : \mathsf{st})$
```

Fig. 11. Ciphertext simulatability game $\mathcal{G}^{\mathsf{SIM\text{-}\$CT}}$

We are now ready to construct private anamorphic signature schemes from SUF-CMA-secure $\varnothing$-recoverable signature schemes. We present the theorem and proof sketch below and provide full details in the full version of this work.

Theorem 8. *Let* S *be a* SUF-CMA*-secure $\varnothing$-recoverable signature scheme and* prE *be a pseudorandom encryption scheme constructed from ideal primitive* Prim *that is simulatable with random ciphertexts. In addition, let* aS = RRep[S, prE]*. Then for all PPT adversaries $\mathcal{A}$ there exist PPT adversaries $\mathcal{B}_0$ and $\mathcal{B}_1$ such that*

$$\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}(O_{\mathsf{LS}},O_{\mathsf{Prog}})}_{\mathsf{aS},\mathsf{Prim},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{SIM\text{-}\$CT}}_{\mathsf{prE},\mathsf{Prim},\mathcal{B}_0}(\lambda) + \mathbf{Adv}^{\mathsf{SUF\text{-}CMA}}_{\mathsf{S},\mathcal{B}_1}(\lambda).$$

S is independent of Prim so the last advantage function does not include it.

Proof Sketch. The proof begins in the game $\mathcal{G}_0 = \mathcal{G}^{\mathsf{RUF\text{-}CAMA}(O_{\mathsf{LS}},O_{\mathsf{Prog}})}$ on aS, where each anamorphic signing query encrypts amsg to obtain act which is used as randomness during the signing process. We transition to a hybrid $\mathcal{G}_1$ where act is instead randomly sampled and SE.Sim is run. Distinguishing $\mathcal{G}_0$ and $\mathcal{G}_1$ is bounded by the SIM-\$CT advantage for SE and Prim. Finally, given an $\mathcal{A}$ against $\mathcal{G}_1$ we construct a PPT $\mathcal{B}_1$ against the SUF-CMA security of S that responds to anamorphic signing queries by querying its own signing oracle, recovering the randomness act, and running SE.Sim. □

Private Anamorphism from Boneh-Boyen Signatures. We briefly discuss Anamorphic Boneh-Boyen signatures proposed in KPPYZ. The Boneh-Boyen signature scheme [7] operates over groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order p and a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Signatures are of the form (r, s) where $r \leftarrow\$ \mathbb{Z}_p$ and s is derived from r and the signing key. The Anamorphic Boneh-Boyen signature scheme simply replaces r with a pseudorandom encryption of the anamorphic message (i.e. RRep). Since Boneh-Boyen is $\varnothing$-recoverable, Anamorphic Boneh-Boyen is private anamorphic (PA-CMA secure).

Studying Anamorphic Boneh-Boyen signatures in the context of our stronger proposed $\mathsf{RUF\text{-}CAMA}$ recipient unforgeability definition, we are unaware of any proof that plain Boneh-Boyen signatures are $\mathsf{SUF\text{-}CRA}$-secure. This is because [7] proves the $\mathsf{SUF\text{-}CMA}$-security of their scheme from a non-adaptive unforgeability notion of an unrandomized scheme where $r = 0$. With (adaptive) chosen randomness, the adversary can query for signatures with randomness $r = 0$.

This gap can be remedied applying our second result regarding RRep. Specifically, [7] introduces a strong Diffie-Hellman assumption and proves that, given this assumption, Boneh-Boyen signatures are $\mathsf{SUF\text{-}CMA}$-secure. We need only instantiate the anamorphic variant aBB with prE satisfying our simulatability property.

Corollary 3. *Consider Boneh-Boyen signature scheme* BB *and let* prE *be an appropriate pseudorandom encryption scheme that is simulatable with random ciphertexts. Then* $\mathsf{aBB} = \mathsf{RRep}[\mathsf{BB}, \mathsf{prE}]$ *is recipient unforgeable provided strong Diffie-Hellman assumption holds.*

5.7 Dictator and Recipient Unforgeable Schemes

We conclude by revisiting the RIdP transform discussed in Sect. 4, which replaces the signing randomness with $r \leftarrow \mathsf{prF}(\mathsf{k}, (\mathsf{ctr}_{\mathsf{Enc}}, \mathsf{msg}, \mathsf{amsg}))$ for a synchronized counter $\mathsf{ctr}_{\mathsf{Enc}}$. Recall that we have already shown that RIdP is stealthy and dictator unforgeable (i.e. a dictator who knows sk cannot adapt or create a new anamorphic signature that contains a valid anamorphic message). We now prove that it produces *recipient* unforgeable anamorphic signature schemes if the underlying plain signature scheme is chosen-randomness secure. As a result, we have shown how to construct anamorphic signature schemes that achieve *all* the unforgeability properties we have proposed.

Theorem 9. *Let* S *be a* $\mathsf{SUF\text{-}CRA}$*-secure signature scheme and* prF *be a pseudorandom function. In addition, let* $\mathsf{aS} = \mathsf{RIdP}[\mathsf{S}, \mathsf{prF}]$ *as described in Construction 2. Then for all PPT adversaries* $\mathcal{A}$ *there exists a PPT adversary* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{RUF\text{-}CAMA}}_{\mathsf{aS},\mathcal{A}}(\lambda) \leq \mathbf{Adv}^{\mathsf{SUF\text{-}CRA}}_{\mathsf{S},\mathcal{B}}(\lambda).$$

Corollary 4. *Consider RSA-PSS from Fig. 1 and let* prF *be an appropriate pseudorandom function. Then anamorphic RSA-PSS derived via* RIdP *i.e.* $\mathsf{aRSA\text{-}PSS} = \mathsf{RIdP}[\mathsf{RSA\text{-}PSS}, \mathsf{prF}]$*, is stealthy (in the* $\mathsf{RoA\text{-}CAMA}$ *sense), dictator unforgeable (in the* $\mathsf{DUF\text{-}CASA}$ *sense), and recipient unforgeable (in the* $\mathsf{RUF\text{-}CAMA}$ *sense).*

We conclude by noting that even though RIdP does not include the signing key sk in the double key dk and all instantiations are private anamorphic as a result, not all instantiations are recipient unforgeable! We give known anamorphic message attacks that recover signing keys (including against RIdP and RIdPX applied to ElGamal) in the full version of this work. In particular, this result shows that there are reasonable instantiations of RIdPX that are *both* robust and private anamorphic, but are *neither* dictator unforgeable (due to the attack we propose in Sect. 4.5) nor recipient unforgeable. This is, active dictators and malicious recipients could easily impersonate parties using RIdPX-derived anamorphic signature schemes even though it was (in essence) intended to be secure against both.

References

1. Ateniese, G., Magri, B., Venturi, D.: Subversion-resilient signatures: Definitions, constructions and applications. Theoretical Computer Science **820**, 91–122 (2020). https://doi.org/10.1016/j.tcs.2020.03.021, https://www.sciencedirect.com/science/article/pii/S0304397520301808
2. Banfi, F., Gegier, K., Hirt, M., Maurer, U., Rito, G.: Anamorphic encryption, revisited. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024, Part II. Lecture Notes in Computer Science, vol. 14652, pp. 3–32. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58723-8_1
3. Bellare, M., Jaeger, J., Kane, D.: Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In: Ray, I., Li, N., Kruegel, C. (eds.) ACM CCS 2015: 22nd Conference on Computer and Communications Security. pp. 1431–1440. ACM Press, Denver, CO, USA (Oct 12–16, 2015). https://doi.org/10.1145/2810103.2813681
4. Bellare, M., Paterson, K.G., Rogaway, P.: Security of symmetric encryption against mass surveillance. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014, Part I. Lecture Notes in Computer Science, vol. 8616, pp. 1–19. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2014). https://doi.org/10.1007/978-3-662-44371-2_1
5. Bellare, M., Rogaway, P.: The exact security of digital signatures: How to sign with RSA and Rabin. In: Maurer, U.M. (ed.) Advances in Cryptology – EUROCRYPT'96. Lecture Notes in Computer Science, vol. 1070, pp. 399–416. Springer, Berlin, Heidelberg, Germany, Saragossa, Spain (May 12–16, 1996). https://doi.org/10.1007/3-540-68339-9_34
6. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) Advances in Cryptology – EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 409–426. Springer, Berlin, Heidelberg, Germany, St. Petersburg, Russia (May 28 – Jun 1, 2006). https://doi.org/10.1007/11761679_25
7. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 56–73. Springer, Berlin, Heidelberg, Germany, Interlaken, Switzerland (May 2–6, 2004). https://doi.org/10.1007/978-3-540-24676-3_4
8. Camenisch, J., Drijvers, M., Gagliardoni, T., Lehmann, A., Neven, G.: The wonderful world of global random oracles. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018, Part I. Lecture Notes in Computer Science, vol. 10820, pp. 280–312. Springer, Cham, Switzerland, Tel Aviv, Israel (Apr 29 – May 3, 2018). https://doi.org/10.1007/978-3-319-78381-9_11

9. Catalano, D., Giunta, E., Migliaro, F.: Anamorphic encryption: New constructions and homomorphic realizations. In: Joye, M., Leander, G. (eds.) Advances in Cryptology – EUROCRYPT 2024, Part II. Lecture Notes in Computer Science, vol. 14652, pp. 33–62. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58723-8_2
10. Craver, S.: On public-key steganography in the presence of an active warden. In: Aucsmith, D. (ed.) Information Hiding. pp. 355–368. Springer Berlin Heidelberg, Berlin, Heidelberg (1998)
11. Degabriele, J.P., Farshim, P., Poettering, B.: A more cautious approach to security against mass surveillance. In: Leander, G. (ed.) Fast Software Encryption – FSE 2015. Lecture Notes in Computer Science, vol. 9054, pp. 579–598. Springer, Berlin, Heidelberg, Germany, Istanbul, Turkey (Mar 8–11, 2015). https://doi.org/10.1007/978-3-662-48116-5_28
12. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology – CRYPTO'84. Lecture Notes in Computer Science, vol. 196, pp. 10–18. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 1984). https://doi.org/10.1007/3-540-39568-7_2
13. Jaeger, J.: Let attackers program ideal models: Modularity and composability for adaptive compromise. In: Hazay, C., Stam, M. (eds.) Advances in Cryptology – EUROCRYPT 2023, Part III. Lecture Notes in Computer Science, vol. 14006, pp. 101–131. Springer, Cham, Switzerland, Lyon, France (Apr 23–27, 2023). https://doi.org/10.1007/978-3-031-30620-4_4
14. Jaeger, J., Tyagi, N.: Handling adaptive compromise for practical encryption schemes. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology – CRYPTO 2020, Part I. Lecture Notes in Computer Science, vol. 12170, pp. 3–32. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 17–21, 2020). https://doi.org/10.1007/978-3-030-56784-2_1
15. Kutylowski, M., Persiano, G., Phan, D.H., Yung, M., Zawada, M.: Anamorphic signatures: Secrecy from a dictator who only permits authentication! In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023, Part II. Lecture Notes in Computer Science, vol. 14082, pp. 759–790. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38545-2_25
16. Kutylowski, M., Persiano, G., Phan, D.H., Yung, M., Zawada, M.: The self-anti-censorship nature of encryption: On the prevalence of anamorphic cryptography. Proc. Priv. Enhancing Technol. **2023**(4), 170–183 (2023). https://doi.org/10.56553/POPETS-2023-0104, https://doi.org/10.56553/popets-2023-0104
17. Mironov, I., Stephens-Davidowitz, N.: Cryptographic reverse firewalls. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology – EUROCRYPT 2015, Part II. Lecture Notes in Computer Science, vol. 9057, pp. 657–686. Springer, Berlin, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46803-6_22
18. Persiano, G., Phan, D.H., Yung, M.: Anamorphic encryption: Private communication against a dictator. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022, Part II. Lecture Notes in Computer Science, vol. 13276, pp. 34–63. Springer, Cham, Switzerland, Trondheim, Norway (May 30 – Jun 3, 2022). https://doi.org/10.1007/978-3-031-07085-3_2

19. Simmons, G.J.: The prisoners' problem and the subliminal channel. In: Chaum, D. (ed.) Advances in Cryptology – CRYPTO'83. pp. 51–67. Plenum Press, New York, USA, Santa Barbara, CA, USA (1983). https://doi.org/10.1007/978-1-4684-4730-9_5
20. Wang, Y., Chen, R., Huang, X., Yung, M.: Sender-anamorphic encryption reformulated: Achieving robust and generic constructions. In: Guo, J., Steinfeld, R. (eds.) Advances in Cryptology – ASIACRYPT 2023, Part VI. Lecture Notes in Computer Science, vol. 14443, pp. 135–167. Springer, Singapore, Singapore, Guangzhou, China (Dec 4–8, 2023). https://doi.org/10.1007/978-981-99-8736-8_5
21. Young, A., Yung, M.: Kleptography: Using cryptography against cryptography. In: Fumy, W. (ed.) Advances in Cryptology – EUROCRYPT'97. Lecture Notes in Computer Science, vol. 1233, pp. 62–74. Springer, Berlin, Heidelberg, Germany, Konstanz, Germany (May 11–15, 1997). https://doi.org/10.1007/3-540-69053-0_6
22. Young, A., Yung, M.: A subliminal channel in secret block ciphers. In: Handschuh, H., Hasan, A. (eds.) SAC 2004: 11th Annual International Workshop on Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 3357, pp. 198–211. Springer, Berlin, Heidelberg, Germany, Waterloo, Ontario, Canada (Aug 9–10, 2004). https://doi.org/10.1007/978-3-540-30564-4_14

Digital Signatures with Outsourced Hashing

Bertram Poettering[1] and Simon Rastikian[1,2](✉)

[1] IBM Research Europe – Zurich, Rüschlikon, Switzerland
[2] ETH Zurich, Zurich, Switzerland
sra@zurich.ibm.com

Abstract. Most practical signature schemes follow the hash-then-sign paradigm: First the (arbitrarily long) message is mapped to a fixed-length hash value, then a signing core derives the signature from the latter. As it is implementationally attractive, practitioners routinely exploit this structure by decoupling the two steps and distributing them among different entities; for instance, industry standards like PKCS#11 specify how security smartcards implement exclusively the core, leaving the hashing to the (untrusted) environment. At the same time, the classic security notions for signature schemes don't consider such a decoupling, and thus don't cover attacks involving, for instance, providing the core with maliciously chosen hash values. A first work that studied this gap appeared only recently (PKC 2024). While it could confirm for a few candidates that they remain secure when split according to PKCS#11, its syntactical abstractions and security definitions are too limited to cover many other practical signature schemes (e.g., the many variants of Fiat–Shamir/Schnorr).

This article studies how the functional separation of hashing and core in signature schemes can be systematized, so that implementational demands (in the spirit of PKCS#11) and, hopefully, security can be met simultaneously. We accompany this foundational work with a case study of a variety of standardized (EC)DLP based signatures. Surprisingly, as we show, their security varies across the full spectrum between universally forgeable and provably unforgeable. For instance, for the same scheme, we demonstrate universal forgeries when instantiated with 224-bit ECC (using an attack that completes in milliseconds), while we establish strong unforgeability for the 256-bit ECC case. Many schemes become completely insecure when the hash function is instantiated with SHA3 instead of with SHA2.

1 Introduction

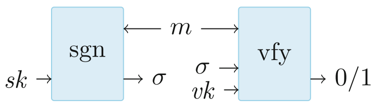


Fig. 1. API of a DSS

The Hash-then-Sign (HtS) Paradigm. Digital signature schemes **(DSS)** provide signers with a mechanism to convince remote verifiers of the authenticity of messages. Most

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15485, pp. 138–167, 2025.
https://doi.org/10.1007/978-981-96-0888-1_5

security protocols depend on a DSS, be it directly (e.g., when signing an email with OpenPGP [13] or X.509 [16]) or indirectly (e.g., in a PKI to authentically distribute long-term public keys for use in some unrelated cryptographic protocol). The central role played by DSS in the cryptographic landscape is also visible in the large number of international standards covering the primitive [4,5,7,8,17,18]. Figure 1 suggests the DSS syntax that we assume in this article.

In order to satisfy the diverse demands of practical settings, the message space $\mathcal{M}$ supported by a DSS should be as general as possible; in practice we can think of $\mathcal{M} = \Sigma^*$ for some convenient alphabet Σ, e.g., $\Sigma = \{0,1\}$ (bits) or $\Sigma = \{0,1\}^8$ (bytes). That is, practical DSS are able to deal with messages of a priori unknown lengths, i.e., they represent variable input-length **(VIL)** primitives. On the other hand, the *core* from which such a DSS is built typically resides in the fixed input-length **(FIL)** domain, i.e., is involved only with constant-size data structures.[1] The technical component bridging the VIL and FIL worlds is typically a cryptographic hash function. The general paradigm behind this is loosely referred to as hash-then-sign **(HtS)**, but details depend on the DSS under consideration. For instance, while in ECDSA a message m is signed by first compressing it down to a hash value $h = H(m)$ and then invoking the DSS core on just h, in Fiat–Shamir inspired signatures the message is hashed as per $h = H(\mathrm{CMT}, m)$ where CMT is the commitment of the underlying Sigma protocol and contributed by the FIL core before the hash value is returned to it. (That is, there is a two-way interaction between the signing core and the hash function.) Intuitively, the above two cases correspond with the algorithm decompositions illustrated in Figs. 2 and 3. We refer to them with the terms Strict HtS and Interactive HtS, respectively.

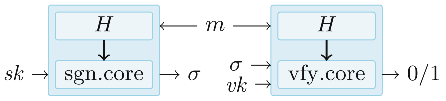


Fig. 2. Strict HtS (intuitively)

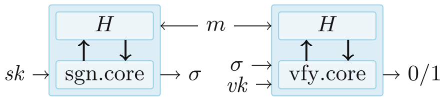


Fig. 3. Interactive HtS (intuitively)

THE (MIS-)USE OF HTS IN PRACTICE. So far we described the use of a hash function to translate the VIL to the FIL setting as a pure construction artifact. In particular, the decomposition of algorithms in the spirit of Figs. 2 and 3 would not be visible at the API level. (The API remains as in Fig. 1). However, *in practice*, (Strict) HtS signature schemes are often used in a different way, e.g., with the core and hashing components implemented at different levels of a software stack or in different pieces of hardware. For instance, if the focus is on effective

[1] For instance, once security parameters are fixed, the field and exponent sizes used in ECDSA are set for good, independently of the lengths of signed messages.

software engineering, the sgn.core algorithm could be implemented in a low-level system library while the H component could be implemented in a high-level language. (Such a separation is often quite natural in practice, as efficiently and safely implementing number-theoretic structures as found in sgn.core typically requires CPU-dependent machine instructions, while hash functions like SHA2 and SHA3 are designed such that they also perform fairly well 'where the data is', i.e., at a higher level of the software hierarchy.) As another example, if implementational security is a priority, the sgn.core component could be implemented in a specially protected smartcard, a hardware security module (HSM), or a trusted platform module (TPM), while the H component could remain implemented in a less trusted environment like a desktop computer. This type of separation is indeed suggested and mandated by the prominent industry standard PKCS#11 that regulates APIs for cryptographic tokens and smartcards: Already in its first version (PKCS#11v1.0, from 1995) [3] it demands that smartcards only implement the signing core of DSA and RSA, rather than the atomic operation that would include the hashing.[2]

Prior Academic Work on HtS Schemes. As the established unforgeability notions for DSS only consider atomic signing, from the perspective of provable security, the decoupled invocation of the subcomponents of an HtS design as described above and suggested by PKCS#11 is by default prohibitive. That is, DSS cannot be assumed safe in the HtS sense until a dedicated analysis confirms their security. To appreciate how security models for HtS have to reach beyond the standard EUF/SUF notions for DSS, consider that one of the above examples specifically assumed that the environment in which H is executed is less trusted than the hardened environment in which sgn.core is operated. An appropriate security model has to reflect this difference of trustworthiness, e.g., by letting the adversary influence, or take control of, the H implementation.[3]

Perhaps surprisingly given the practical relevance of HtS schemes, formal HtS security definitions and analyses have been lacking so far, with the one recent exception of [21] (PKC 2024) which exclusively covers the Strict HtS case (Fig. 2). In a nutshell, [21] studies (and confirms) the security of a number of common (EC)DLP-based HtS signatures in a setting where the adversary controls the hash function implementation.[4] The Interactive HtS case (Fig. 3) is not considered in [21].

[2] See, for instance, "This [CKM_DSA] mechanism corresponds only to the part of DSA that processes the [SHA1] hash value; it does not compute the hash value." in [3, Sect. 10.6] or "This [RSA] mechanism corresponds only to the part [...] that involves RSA; it does not compute a message digest" in [3, Sect. 10.2].

[3] Indeed, [21] reports on a DSS of the Strict HtS type (of Fig. 2) that is, when used atomically, unforgeable in the classic sense, but trivially fails if an adversary controls the H implementation.

[4] Specifically, the analyzed schemes are ECDSA and its Russian GOST and Chinese SM2 counterparts.

1.1 Contributions

As already noted: While the HtS structure of signatures is regularly exploited in practice, and is even suggested by relevant industry standards, corresponding academic support is, so far, restricted to results that consider the Strict HtS case (Fig. 2, [21]). Observing that modern DSS proposals (e.g., EdDSA [18], ECSDSA [5], ML-DSA/Dilithium [9], Falcon [6]) are *not* Strict HtS but Interactive HtS schemes (Fig. 3), in this work we first lay the formal foundations of the latter class of schemes, and then verify for a number of Interactive HtS candidates whether they meet these notions. In more detail, we first propose a syntax ("API") for Interactive HtS schemes, then develop and study suitable security notions, and finally provide security proofs for some HtS candidates while demonstrating attacks on others. We discuss these contributions one by one in the following.

Syntax Definition. We start with introducing the DSS with Outsourced Hashing **(DSSwOH)** syntactical abstraction of a DSS where hashing is made explicit. (The subtle difference between DSSwOH and Interactive HtS will become clear in the subsequent paragraph.) As illustrated in Fig. 4, we model the signing core as the (stateful) consecutive execution of first an initialization and then a finalization algorithm (sgn.ini → sgn.fin): The initialization algorithm creates a hash seed *hs* for the hashing algorithm *hash* which translates hash seed and message into a hash value *hv* that is consumed by the finalization algorithm to generate the signature. The verifier is modeled analogously.[5]

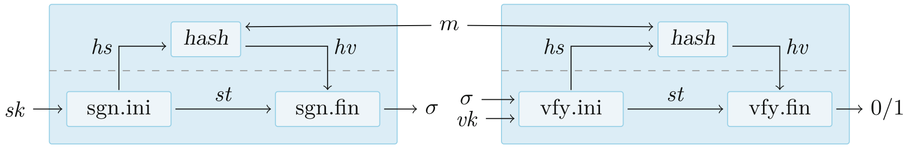


Fig. 4. Illustration of the DSSwOH syntax. Note the outer API is as in Fig. 1. The horizontal dashed lines illustrate the conceptual separation of the message hashing entity from the signing/verification core.

Importantly, and in contrast with what Fig. 3 might suggest, our DSSwOH abstraction does *not* demand that the boundaries between core algorithms and hashing necessarily coincide with where the operations of, say, SHA2 or SHA3, begin and end. Our syntax is more general than that, as is best illustrated at hand of an example. (The following high-level exposition is tuned towards accessibility, brushing off a large number of details; see the paper body, in particular Sect. 2.4

[5] As signature verification is a public operation, it is security-wise less relevant whether its core and hashing are implemented atomically or separately. While the focus of this work is hence on the signing part, it is conceptionally and notationally convenient to allow also verifiers to separate-off their hashing.

and Fig. 9, for precise definitions.) Consider the case of Schnorr signatures where a signature consists of a pair $\sigma = (R, s)$ satisfying $g^s = RX^{H(R,m)}$, with X representing the verification key and H a hash function. Assume H is instantiated with a Merkle–Damgård construction (like SHA2) which computes hash values by processing its inputs online, i.e., in a stateful left-to-right fashion. It is then implied that the value $H(R, m)$ can be computed, iteratively in three stages, as per $H(R, m) = h_3(h_2(h_1(R), m))$ for suitably defined functions h_1, h_2, h_3. In this setting, a natural and promising way to fit the Schnorr DSS into our DSSwOH syntax is as follows: The sgn.ini algorithm prepares signature value R and outputs $hs = h_1(R)$ as the hash seed; the *hash* algorithm proceeds with computing $hv = h_2(hs, m)$; the sgn.fin algorithm completes the hash function evaluation by computing $h_3(hv)$, obtaining the value $H(R, m)$ required to derive the signature σ. Note that this way of distributing the Merkle–Damgård iterations of a single hash function evaluation among the three sgn.ini, *hash*, sgn.fin algorithms is just one way to consider the Schnorr DSS in the DSSwOH framework; other interesting options might exist.

Security Definitions. Now that we have fixed a DSSwOH syntax, the next step is to develop suitable security definitions. Our specifications are game-based and extend the standard EUF/SUF notions (and the notions introduced in [21]) by new concepts along two different dimensions: **(1)** Our models make the two-way interaction between the core operations and the hashing in the spirit of Fig. 3 explicit; so that we are able to analyze cases where the entity performing the hashing is less trusted than the core entity (desktop computer vs. smartcard), some variants of our games assume that the *hash* component is fully controlled by the adversary; **(2)** As signing and verifying are now stateful operations (see Fig. 4), aspects like the concurrent invocation of several sgn.ini $\rightarrow$ *hash* $\rightarrow$ sgn.fin sessions have to be taken into account; our games hence implement a session concept and allow the adversary to interact with multiple parallel sessions.

At the formal level, we introduce a family of security games, double-subindexed by indications of the considered hashing trustworthiness for the signer and verifier, respectively. For instance, our unforgeability game $\mathbf{UF}^{\mathrm{e}}_{\mathrm{MB}}$ models existential unforgeability (superindex ‘e’) where the signer’s hashing implementation is malicious (first subindex ‘M’) and the verifier’s hashing implementation is benign (second subindex ‘B’). Here, hashing is considered malicious if the adversary fully controls the hash value computation, while hashing is benign if the (passive) adversary just sees, but cannot tamper with, the externalized hashing operations. In addition to ‘M’ and ‘B’, we use the symbol ‘A’ to denote the atomic case where signing (resp. verifying) is executed as a single operation without involving the adversary. Note the $\mathbf{UF}^{\mathrm{e}}_{\mathrm{AA}}$ game coincides with the classic EUF-CMA definition for DSS. As the A/B/M variants form a sequence of increasing strength, overall we obtain a lattice of implications as indicated in Fig. 5 (a).

Relations Among Security Notions. We make two general observations on the relations between the notions of Fig. 5 (a): **(1)** For DSS with a deterministic verification algorithm (i.e., for *all* standardized DSS), we have that $x\mathrm{A} \Rightarrow x\mathrm{B}$ for

$$\begin{tikzcd}
 & & MM \arrow[dl] \arrow[dr] & & \\
 & MB \arrow[dl] \arrow[dr] & & BM \arrow[dl] \arrow[dr] & \\
MA \arrow[dr] & & BB \arrow[dl] \arrow[dr] & & AM \arrow[dl] \\
 & BA \arrow[dr] & & AB \arrow[dl] & \\
 & & AA & &
\end{tikzcd}$$

(a) Lattice of security notions.

$$\begin{tikzcd}
 & & MM \arrow[dl] \arrow[dr] & & \\
 & MB \arrow[dl, leftrightarrow] \arrow[dr] & & BM \arrow[dl] \arrow[dr] & \\
MA \arrow[dr] & & BB \arrow[dl, leftrightarrow] \arrow[dr] & & AM \arrow[dl] \\
 & BA \arrow[dr] & & AB \arrow[dl, leftrightarrow] & \\
 & & AA & &
\end{tikzcd}$$

(b) Fiat–Shamir lattice.

$$\begin{tikzcd}
MA \arrow[d] \\
BA \arrow[d] \\
AA
\end{tikzcd}$$

(c) FS

Fig. 5. Left: Implications between different levels of hashing trustworthiness, where $\mathbf{UF}^{e}_{\mathrm{MM}}$ is the strongest notion and $\mathbf{UF}^{e}_{\mathrm{AA}}$ is the weakest. **Middle:** Sublattice relevant for Fiat–Shamir based signatures. **Right:** Compact version of (b).

any $x \in \{\mathrm{A}, \mathrm{B}, \mathrm{M}\}$, meaning that if a DSSwOH is secure according to game $\mathbf{UF}_{x\mathrm{A}}$ then it is also secure according to game $\mathbf{UF}_{x\mathrm{B}}$. This is so because oracle access to a deterministically evaluated public function is simulatable by the adversary, i.e., reveals no information.[6] **(2)** No DSSwOH derived via the Fiat–Shamir transform from a ZK proof system can reach MM or BM or AM security. By Fig. 5 (a) we only need to argue for the AM case where the following blueprint lays the basis of an attack on $\mathbf{UF}_{\mathrm{AM}}$ that always succeeds (and doesn't even require a signing query): The adversary first invokes the ZK simulator to obtain a valid ZK transcript $(\mathrm{CMT}, \mathrm{CH}, \mathrm{RSP})$, then exploits the control over the verifier's hashing algorithm to ensure that $\mathrm{CMT} \parallel m^*$ maps to CH, where m^* is the message forged on.

An important class of standardized DSS are instantiations of the scheme of Schnorr [23]. These meet the conditions of (1) and (2) above, meaning that, for them, the lattice of Fig. 5 (a) collapses to the lattice of Fig. 5 (b), which can be written more compactly as in Fig. 5 (c). To conclude: The three most relevant security notions considered in this article are $\mathrm{AA}, \mathrm{BA}, \mathrm{MA}$ (sorted by increasing strength), where the first coincides with the standard EUF-CMA notion, the second outsources the hashing of the signer but doesn't allow tampering with it, and the third assumes a malicious implementation of the signer's hashing routine.

Analyses of Existing DSS. For a set of standardized DSS we study whether they retain security when operated with outsourced hashing, i.e., as a DSSwOH. Specifically we analyzed the schemes SDSA [5], ECSDSA [4], ECSDSA Optimized [5], BIP 340 [25], and ECFSDSA [5], which have been standardized by different international bodies, and which all can be seen as real-world instantiations of the Schnorr signature scheme [23], but with details implemented slightly

[6] By Fig. 5 (a) we also have $x\mathrm{B} \Rightarrow x\mathrm{A}$, meaning that deterministic verification actually implies $x\mathrm{A} \Leftrightarrow x\mathrm{B}$.

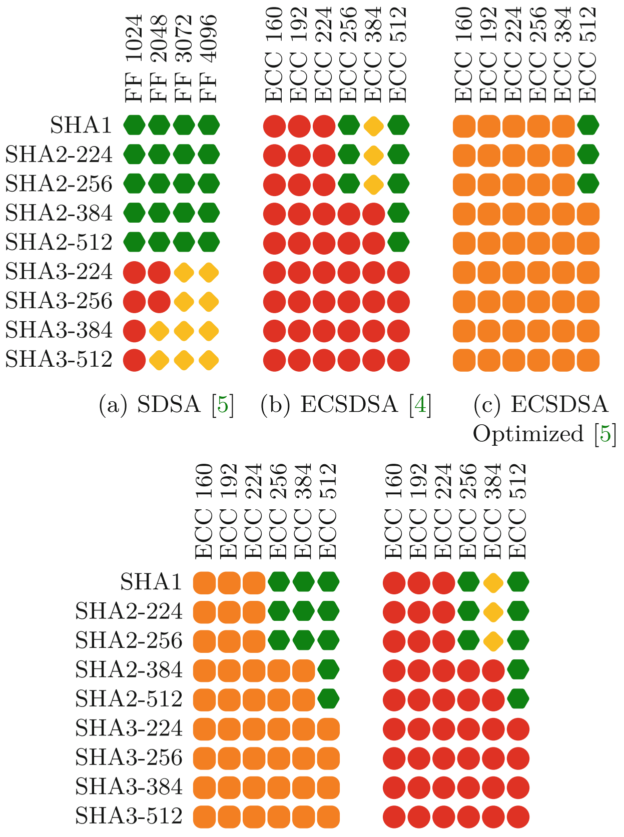
differently.[7,8] Our analyses start with formatting the schemes according to the DSSwOH syntax, where we exploit, in the way suggested towards the beginning of Sect. 1.1, that the approved hash functions of all standards are iterated, i.e., are either Merkle–Damgård or sponge based.

We then turn to security analyses in our models. (Recall from above that in the case of Schnorr signatures the pivotal notions are BA and MA, see also Fig. 5c.) While one could expect a certain level of similarity among the results for each scheme—at the end of the day, all are Schnorr-based!— the security profiles turn out to be largely diverse. Figure 6 provides a first overview, where a ⬢ mark indicates that a variant is provably secure (even in the strong MA sense), a ● mark indicates that a variant is insecure (with an efficient universal forgery attack, even in the weak BA sense), a ◆ mark indicates that it is unlikely that one can prove security (but we don't have an attack), and a ■ mark indicates that a candidate is 'almost broken' (our attacks don't apply, but it is conceivable to likely that simple variations would do). As visible in Fig. 6, whether a specific candidate DSSwOH turns out to be secure or forgeable critically depends on the choice of underlying cyclic group and hash function. For instance, we present universal attacks on ECSDSA signatures when instantiated with SHA256 and 160-bit, 192-bit, or 224-bit ECC, while we prove security for 256-bit and 512-bit ECC. If SHA256 is replaced by SHA3, all ECC choices are insecure.

Details on Attacks/Proofs. We provide insight on why the security status of fairly similar configurations of a single scheme may vary so strongly (from universally forgeable to provably secure). Our attacks make use of three properties that are specific to the DSSwOH setting: **(1)** The signing core publicly commits to the hash seed before the adversary commits to the message. (See in Fig. 4 that every signing operation starts with sgn.ini outputting hs, while m is required only later.) **(2)** In the case of Fiat–Shamir signatures (in particular: Schnorr signatures), the hash function is evaluated as per $H(\mathrm{CMT}, m)$, where CMT is the ZK commitment. That is, necessarily, value CMT will in some form be reflected in hash seed hs, and the *hash* algorithm will only 'add' the message m. Assume for now that CMT can always be extracted from hs. **(3)** As argued above, concurrent signing sessions have to be assumed in the DSSwOH setting. (That is, in the terms of Fig. 4, there may be multiple independent states st at the same time.) Now, as worked out in [11], if the ZK-commitments of concurrent interactive Schnorr signing sessions are known before the messages

[7] The five standards are pairwise incompatible with each other, and differ with respect to at least the following: The groups and hash functions that may be used; whether point compression is used and how; whether public keys, or even PKI certificates, are included in the hash operation.

[8] Regarding BIP 340 we observe that [25] specifies a number of different options to implement the signing algorithm; some of these are randomized and others are deterministic. In this work we only analyze the fully randomized case. As the deterministic (derandomized) case requires two hashing passes over the message, it is not compatible with our DSSwOH abstraction.



	FF 1024	FF 2048	FF 3072	FF 4096
SHA1	🟢	🟢	🟢	🟢
SHA2-224	🟢	🟢	🟢	🟢
SHA2-256	🟢	🟢	🟢	🟢
SHA2-384	🟢	🟢	🟢	🟢
SHA2-512	🟢	🟢	🟢	🟢
SHA3-224	🔴	🔴	🔶	🔶
SHA3-256	🔴	🔴	🔶	🔶
SHA3-384	🔴	🔶	🔶	🔶
SHA3-512	🔴	🔶	🔶	🔶

(a) SDSA [5]

	ECC 160	ECC 192	ECC 224	ECC 256	ECC 384	ECC 512
SHA1	🔴	🔴	🔴	🟢	🔶	🟢
SHA2-224	🔴	🔴	🔴	🟢	🔶	🟢
SHA2-256	🔴	🔴	🔴	🟢	🔶	🟢
SHA2-384	🔴	🔴	🔴	🔴	🔴	🟢
SHA2-512	🔴	🔴	🔴	🔴	🔴	🟢
SHA3-224	🔴	🔴	🔴	🔴	🔴	🔴
SHA3-256	🔴	🔴	🔴	🔴	🔴	🔴
SHA3-384	🔴	🔴	🔴	🔴	🔴	🔴
SHA3-512	🔴	🔴	🔴	🔴	🔴	🔴

(b) ECSDSA [4]

	ECC 160	ECC 192	ECC 224	ECC 256	ECC 384	ECC 512
SHA1	🟧	🟧	🟧	🟧	🟧	🟢
SHA2-224	🟧	🟧	🟧	🟧	🟧	🟢
SHA2-256	🟧	🟧	🟧	🟧	🟧	🟢
SHA2-384	🟧	🟧	🟧	🟧	🟧	🟧
SHA2-512	🟧	🟧	🟧	🟧	🟧	🟧
SHA3-224	🟧	🟧	🟧	🟧	🟧	🟧
SHA3-256	🟧	🟧	🟧	🟧	🟧	🟧
SHA3-384	🟧	🟧	🟧	🟧	🟧	🟧
SHA3-512	🟧	🟧	🟧	🟧	🟧	🟧

(c) ECSDSA Optimized [5]

	ECC 160	ECC 192	ECC 224	ECC 256	ECC 384	ECC 512
SHA1	🟧	🟧	🟧	🟢	🟢	🟢
SHA2-224	🟧	🟧	🟧	🟢	🟢	🟢
SHA2-256	🟧	🟧	🟧	🟢	🟢	🟢
SHA2-384	🟧	🟧	🟧	🟧	🟧	🟢
SHA2-512	🟧	🟧	🟧	🟧	🟧	🟢
SHA3-224	🟧	🟧	🟧	🟧	🟧	🟧
SHA3-256	🟧	🟧	🟧	🟧	🟧	🟧
SHA3-384	🟧	🟧	🟧	🟧	🟧	🟧
SHA3-512	🟧	🟧	🟧	🟧	🟧	🟧

(d) BIP 340 [25]

	ECC 160	ECC 192	ECC 224	ECC 256	ECC 384	ECC 512
SHA1	🔴	🔴	🔴	🟢	🔶	🟢
SHA2-224	🔴	🔴	🔴	🟢	🔶	🟢
SHA2-256	🔴	🔴	🔴	🟢	🔶	🟢
SHA2-384	🔴	🔴	🔴	🔴	🔴	🟢
SHA2-512	🔴	🔴	🔴	🔴	🔴	🟢
SHA3-224	🔴	🔴	🔴	🔴	🔴	🔴
SHA3-256	🔴	🔴	🔴	🔴	🔴	🔴
SHA3-384	🔴	🔴	🔴	🔴	🔴	🔴
SHA3-512	🔴	🔴	🔴	🔴	🔴	🔴

(e) ECFSDSA [5]

Fig. 6. Legend: 🔴: broken 🟧: almost broken 🟢: secure 🔶: less secure. Security of configurations of five standardized DLP-based signature schemes.

need to be committed to, then forgeries are efficiently possible.[9] These forgeries also represent forgeries in the DSSwOH setting.

A prerequisite of the above attack is that the ZK-commitment CMT can always be extracted from hs. Whether this condition is actually fulfilled very much depends on the hash function used. For instance, the internals of SHA3-256 are such that the sponge permutation is executed once every 1088 bits of input. If a Schnorr signature like ECSDSA is instantiated with 256-bit ECC, then the ZK-commitment will be at most $2 \cdot 256 = 512$ bit in length (whether point compression is used or not). That is, as $512 \ll 1088$, the sponge permutation will not have been executed when hs was prepared, meaning that CMT can indeed

[9] The language and attacks of [11] are not as generic as we need them to be to achieve our results; hence, in Sect. 5 we first generalize and re-prove the findings of [11] before we use them in our attacks.

be extracted from its internal buffer, ultimately leading to a successful forgery attack. All the 🔴 marks of Fig. 6 result from an argument in this spirit.

But note that not in all cases it is possible to extract CMT from hs. For instance, consider that SHA2-256 invokes its compression function once every 512 bits, and the compressed (respectively, uncompressed) encoding length of 512-bit ECC curve points is 512 bits (resp. $2 \cdot 512 = 1024$ bits). In this case, hs is precisely the output of a (double) evaluation of the Merkle–Damgård compression function on CMT, practically meaning that no information on CMT can be extracted from hs. In fact, as we show, this is sufficient to make a formal security argument possible. All ⬢ marks of Fig. 6 result from an argument in this spirit. (The remaining ◆/🟧 marks result from partial-extraction cases, see body of the article.)

1.2 Motivation and More Related Work

The Hash-then-Sign paradigm is folklore among theoreticians and practitioners. On the academic side, while textbooks regularly state and prove that hashing a message and then signing the hash value results in a secure DSS, the focus is never specifically on outsourcing hashing to a separate entity, e.g., to fit a smartcard setting.

In industry, the advantages of outsourced hashing have been recognized early, see [22]. Strict HtS schemes like DSA, ECDSA and PKCS#1 v1.5 are naturally attractive for outsourcing the hashing, simply as a DSSwOH implementation wouldn't leak information to the adversary. Common cryptographic libraries such as BoringSSL [15], PyCryptoDome [1] and Gcrypt [14] implement these schemes with the option to outsource the hashing. Industry discussions in the context of the standardization of post-quantum replacements of (EC)DSA also pointed out that these replacement ideally would be Strict HtS (but likely will not be) [19,20,24]. Strict HtS schemes (ECDSA along with the Russian, Chinese and Korean analogues) have been previously covered in [21].

The dominant industry standard in the domain of HSMs, security tokens, and smartcards is PKCS#11 [26]. We note that this standard explicitly defines signature mechanisms where hashing is removed from the signing procedure and outsourced to an external entity, e.g. under the name of "raw RSA" [26, Sect. 2.2.12], "DSA without hashing" [26, Sect. 2.2.11], or "ECDSA without hashing" [26, Sect. 2.3.12]. The standard makes no statements about the security of these schemes and the required trustworthiness of the hashing implementation.

2 Preliminaries

We expand in Appendix A on the vertical arrows used in some of the pseudo-code figures.

2.1 Notation

When, in this article, we use pseudo-code, ‘ $\leftarrow$ ’ refers to the assign operator, and ‘ $\xleftarrow{\$}$ ’ refers to a (uniformly) random assignment. We use the symbol ‘ ϵ ’ to denote the empty string and ‘ $:=$ ’ as an alias-creating operator: The instruction $x := A$ introduces x as a symbolic alias for the expression A. When x is assigned any value, A is automatically assigned this same value, and vice versa. We write $y \leftarrow \mathrm{alg}\langle st\rangle(x)$ for the case where algorithm alg takes (its state st and) x on input, updates its internal state st, and outputs y.

In our figures, variables written in capital letters (roman font) generally denote arrays or sets depending on the context, and elements in arrays are designated by an index between square brackets. We use the diamond $\diamond$ as a special character that symbolizes the empty element and the dot in brackets $\mathrm{A}[\cdot]$ to denote all the elements in an array A. If X and Y are two sets, then we denote $\mathrm{X} \xleftarrow{\cup} \mathrm{Y}$ shorthand for $\mathrm{X} \leftarrow \mathrm{X} \cup \mathrm{Y}$.

We formalize security properties with games written in pseudo-code. These games invoke an efficient adversary $\mathcal{A}$ that may be provided with access to oracles. The games terminate when executing *Stop with* $\cdot$ command where $\cdot$ represents a Boolean constant. We write $\Pr[G(\mathcal{A})]$ for the probability that game G invoked with adversary $\mathcal{A}$ stops with $\top$. We call this probability the accepting probability. We introduce the instructions *Lose* as a shorthand for ‘Stop with $\bot$’, *Require* $\cdot$ which stands for ‘if not $\cdot$ then Lose’ and *Promise* $\cdot$ to summarize ‘if not $\cdot$ then Stop with $\top$’. In some games, we use the word *Share* to mean that the next variable is made public (particularly delivered to the adversary) without returning the overall algorithm. A cryptographic scheme algorithm may fail or *Abort*, in particular, if a scheme algorithm aborts, then the corresponding oracle immediately aborts as well returning to the adversary.

2.2 Hash Function APIs

ITERATIVE HASHING. Practical cryptographic hash functions $\{0,1\}^* \rightarrow \{0,1\}^l$ handle variable-length inputs by feeding them into an internal iteratively-invoked fixed-length building block. For Merkle–Damgård **(MD)** constructions like SHA1 and the members of the SHA2 family, this building block is a compression function CF that maps a κ-bit chaining value and an ρ-bit input block to an updated κ-bit chaining value. For sponge-based hash functions like the members of the SHA3 family, this building block is a permutation Π that maps a ρ-bit input block (‘rate’) and a κ-bit chaining value (‘capacity’) to a ρ-bit output block and an updated κ-bit chaining value. In both cases, quantities κ and ρ are specific to the hash function. Quantity ρ is also referred to as the hash function’s *block size* as it indicates the number of input bits that are compressed at a time. Practical values of ρ range from 64 bytes to 144 bytes; see the full version for an overview.

STREAM-ORIENTED APIS. Despite internally working with constant-length data units, the typically implemented API of a hash function is not block-oriented but stream-oriented: After initializing an instance state, applications advance the hash computation by serially providing arbitrary-length fragments of the message. The hash function implementation will then arrange the fragments into ρ-bit blocks and apply the fixed-length primitive (i.e., compression function or permutation) to only those. This stream-oriented approach fully hides the (purely technical) block size ρ from the application, ensuring versatility and cryptographic agility.

In Fig. 7 we reproduce a typical API and implementation of an MD hash function. (The very same principle as described here also holds for sponge based hash functions; just details differ, for instance how the final message digest is computed. See also the full version.) While the procedure names and their precise semantics are based on those of the Python standard library [2], the same approach is used in virtually all other SHA1, SHA2, and SHA3 implementations as well. Invoking procedure *new* initiates a new hash value computation by creating a fresh state *st* which consists of three components: the chaining value *cv* (initialized to some constant value IV), a ρ-bit string buffer B (initialized to ε), and the number *cnt* of processed bits (initialized to zero). Input fragments m can then be fed into the hash state by invoking the *update* procedure. The latter is concerned with arranging the fragments into a sequence of full blocks, and invoking the compression function CF as required. Note that, at any time, for any two message fragments m_1, m_2, invoking $update\langle st\rangle(m_1)$ and $update\langle st\rangle(m_2)$ in direct succession has precisely the same effect as the single-shot invocation $update\langle st\rangle(m_1 \| m_2)$. To conclude a hash computation, procedure *digest* appends a representation of the overall message length to the message, invoking the compression function whenever required, and outputs (a possibly truncated copy of) the last chaining value.

```
Proc. $new$
00 $cv \leftarrow \mathrm{IV}$
01 $B \leftarrow \epsilon$
02 $cnt \leftarrow 0$
03 $st := (cv, B, cnt)$
04 Return $st$

Proc. $update\langle st\rangle(m)$
05 $cnt \leftarrow cnt + |m|$
06 If $|B \| m| < \rho$:
07     $B \leftarrow B \| m$
08 Else:
09     $m' \| m \leftarrow m$
10     $cv \leftarrow \mathrm{CF}(cv, B \| m')$
11     While $|m| \geq \rho$:
12         $m' \| m \leftarrow m$
13         $cv \leftarrow \mathrm{CF}(cv, m')$
14     $B \leftarrow m$

Proc. $digest\langle st\rangle$
15 $\bar{m} \leftarrow pad(cnt)$
16 If $|B \| \bar{m}| = 2 \cdot \rho$:
17     $B_1 \| B_2 \leftarrow B \| \bar{m}$
18     $cv \leftarrow \mathrm{CF}(cv, B_1)$
19     $cv \leftarrow \mathrm{CF}(cv, B_2)$
20 Else:
21     $cv \leftarrow \mathrm{CF}(cv, B \| \bar{m})$
22 $h \leftarrow trunc(cv)$
23 Return $h$
```

Fig. 7. Blueprint of stream-oriented Merkle–Damgård implementations. Line 09 splits m into m' and (updated) m such that $|B \| m'| = \rho$. Line 12 splits m into m' and (updated) m such that $|m'| = \rho$. Length padding $\bar{m}$ in line 15 is such that $|B \| \bar{m}| \in \{\rho, 2 \cdot \rho\}$. Line 17 splits $B \| \bar{m}$ such that $|B_1| = |B_2| = \rho$.

Message Extraction from Hash States. In the context of this article, a crucial property of stream-oriented hash function implementations (both MD and sponge based) is that message fragments are buffered in the hash instance state and may remain recoverable from it. This is visible in lines 07 and 14 of Fig. 7 where message inputs are recorded in B but are not cryptographically processed. For instance, for any message m of size one bit shorter than a multiple of the block size, i.e., with $|m|+1 \equiv 0 \pmod{\rho}$, after $st \leftarrow \mathit{new}()$; $\mathit{update}\langle st\rangle(m)$ is invoked, the first $\rho - 1$ bits of B coincide with the last $\rho - 1$ bits of m.

The above is an unavoidable property of MD implementations. In contrast, for sponge based hash functions like SHA3 the situation might be slightly different, as for the latter the incoming message fragments are not necessarily buffered but instead may be XORed into the chaining value cv. That is, unless the output of the last Π invocation is known, only masked versions of the message fragments can be extracted from the hash state st. We note, however, that common SHA3 implementations don't XOR directly but instead follow a buffer-then-XOR schema. This is, for instance, the case for OpenSSL's SHA3 implementation which serves also as the default SHA3 implementation of many high-level languages.[10] We refer the reader to the full version for more details about MD based and sponge-based constructions.

Partial Hashing and Random Oracles. The observed extractability property holds for message fragments that are buffered and thus not cryptographically processed, and hence remain extractable from the hashing state. For message parts that *are* cryptographically processed, a complementary observation can be made: The hashing state hides them in an ideal way (if the compression function behaves like a random oracle). In more detail: Assuming an MD hash function, for any multi-block message $m = m_1 \parallel \ldots \parallel m_n$ consider the state $st = (cv_m, B_m, cnt_m)$ resulting from invoking first $st \leftarrow \mathit{new}()$ and then $\mathit{update}\langle st\rangle(m)$. Let $H_n\colon \{0,1\}^{n\rho} \to \{0,1\}^{\kappa};\ m \mapsto cv_m$ be the function induced by this, i.e., that maps messages to corresponding intermediate chaining values. Then, for any fixed n, if the compression function $\mathrm{CF}\colon \{0,1\}^{\kappa} \times \{0,1\}^{\rho} \to \{0,1\}^{\kappa}$ is modeled as a random oracle, then H_n is (indifferentiable from) a random oracle. We formally state this in Lemma 2.

Note that a corresponding statement cannot be made for sponge constructions. The reason is that the permutation Π is invertible and hence doesn't hide its input. Concretely, if cv_m of an unknown one-block message m is known, then computing $\Pi^{-1}(cv_m)$ immediately recovers m which is a property incompatible with a random oracle.

[10] OpenSSL's SHA3 code can be found at https://github.com/openssl/openssl/blob/1c0eede9/crypto/sha/sha3.c. It precisely follows the blueprint of Fig. 7. Python uses a copy of this code since version 3.9, see "Hashlib [. . .] uses SHA3 [. . .] from OpenSSL" in https://docs.python.org/3/library/hashlib.html.

2.3 Signature Schemes

We recall the established notions of DSS. Our definitions of existential and strong unforgeability (UF^{e} and UF^{s}) are equivalent with the textbook notions, but, for alignment with the peculiarities of the generalized DSS that we study in the upcoming sections, we use a slightly unusual game notation that assumes an explicit verification oracle.

DSS. A *digital signature scheme* **(DSS)** for a message space $\mathcal{M}$ consists of a signing key space $\mathcal{SK}$, a verification key space $\mathcal{VK}$, a signature space $\mathcal{S}$, a key generation algorithm $\mathrm{gen} \to \mathcal{SK} \times \mathcal{VK}$, a signing algorithm $\mathcal{SK} \times \mathcal{M} \to \mathrm{sgn} \to \mathcal{S}$, and a verification algorithm $\mathcal{VK} \times \mathcal{M} \times \mathcal{S} \to \mathrm{vfy} \to \{0,1\}$. Intuitively, for correctness we expect that for all $m \in \mathcal{M}$ after $(sk, vk) \leftarrow \mathrm{gen}$ and $\sigma \leftarrow \mathrm{sgn}(sk, m)$ and $v \leftarrow \mathrm{vfy}(vk, m, \sigma)$ we have $v = 1$.

Definition 1 (Unforgeability). *Consider the* $\mathbf{UF}^{\mathrm{e}}$ *and* $\mathbf{UF}^{\mathrm{s}}$ *games of Fig. 8. We define unforgeability advantages of an adversary* $\mathcal{A}$ *as per* $\mathbf{Adv}^{\text{uf-e}}(\mathcal{A}) := \Pr[\mathbf{UF}^{\mathrm{e}}(\mathcal{A})]$ *and* $\mathbf{Adv}^{\text{uf-s}}(\mathcal{A}) := \Pr[\mathbf{UF}^{\mathrm{s}}(\mathcal{A})]$*, respectively. Intuitively, we say that a DSS is existentially (respectively, strongly) unforgeable if* $\mathbf{Adv}^{\text{uf-e}}(\mathcal{A})$ *(resp.,* $\mathbf{Adv}^{\text{uf-s}}(\mathcal{A})$*) is negligible for all realistic* $\mathcal{A}$*.*[11]

```
Initializations: $\mathrm{A}, \mathrm{A}^{\cdot}, \mathrm{V}, \mathrm{V}^{\cdot} \leftarrow \emptyset$

Game $\mathbf{UF}^{\mathrm{e,s}}(\mathcal{A})$
00    $sk, vk \leftarrow \mathrm{gen}$
01    $\mathcal{A}(vk)$
02^e  Promise $\mathrm{V} \subseteq \mathrm{A}$
03^s  Promise $\mathrm{V}^{\cdot} \subseteq \mathrm{A}^{\cdot}$
04    Lose

Oracle Sgn$(m)$
05    $\sigma \leftarrow \mathrm{sgn}(sk, m)$
06    Share $\sigma$
07    $\mathrm{A} \overset{\cup}{\leftarrow} \{m\}$
08    $\mathrm{A}^{\cdot} \overset{\cup}{\leftarrow} \{(m, \sigma)\}$

Oracle Vfy$(m, \sigma)$
09    $v \leftarrow \mathrm{vfy}(vk, m, \sigma)$
10    Share $v$
11    If $v$: $\mathrm{V} \overset{\cup}{\leftarrow} \{m\}$
12    If $v$: $\mathrm{V}^{\cdot} \overset{\cup}{\leftarrow} \{(m, \sigma)\}$
```

Fig. 8. Games $\mathbf{UF}^{\mathrm{e}}, \mathbf{UF}^{\mathrm{s}}$ for Definition 1: Line 02 is part of $\mathbf{UF}^{\mathrm{e}}$ but not of $\mathbf{UF}^{\mathrm{s}}$, while line 03 is part of $\mathbf{UF}^{\mathrm{s}}$ but not of $\mathbf{UF}^{\mathrm{e}}$. Set variable A records the authentic messages (i.e., considered by the signer) while set variable V records the verified messages (i.e., accepted by the verifier). By line 02 the adversary wins the $\mathbf{UF}^{\mathrm{e}}$ game if there exists a verified message that is not authentic. The winning condition of the $\mathbf{UF}^{\mathrm{s}}$ game is similar, just that it is based on message-signature pairs which are recorded in the punctured sets $\mathrm{A}^{\cdot}, \mathrm{V}^{\cdot}$.

[11] Practical DSS don't have a security parameter 'λ' that could scale security arbitrarily, and hence an asymptotic notion of a 'ppt adversary' is not meaningful for their analysis. Our use of the word 'realistic adversary' can be seen as an intuitive replacement of the ppt notion. We only use it in informal statements and hence don't need to defined it.

2.4 Schnorr Proofs, Schnorr Signatures

Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of prime order p. Denote the 'exponent space' with $\mathbb{Z}_p$. Consider $\mathbb{G}$ and $\mathbb{Z}_p$ a statement and witness space, respectively, such that x acts as the witness for $X = g^x$. The interactive ZK-PoK proof system by Schnorr lets the prover compute a commitment R as per $R \leftarrow g^r$ (for a uniform $r \in \mathbb{Z}_p$), lets the verifier pick a high-entropy challenge c, and lets the prover compute a response s as per $s \leftarrow r + xc$. The verifier accepts the resulting transcript $\tau = (R, c, s)$ iff $g^s = RX^c$. Applying the Fiat–Shamir transform, Schnorr signatures assume a hash function $H \colon \mathbb{G} \times \mathcal{M} \to \mathbb{Z}_p$, let $c := H(R, m)$, and use a compressed version of the transcript τ as the signature σ. More precisely, instead of letting $\sigma := \tau$, all standardized versions of Schnorr DSS let $\sigma = (c, s)$ (equivalently $\sigma = (R, s)$) as τ can be uniquely reconstructed from σ and public information. The resulting DSS algorithms are in Fig. 9 (left).

```
Proc. gen
00 $x \xleftarrow{\$} \mathbb{Z}_p$
01 $X \leftarrow g^x$
02 $sk := x$
03 $vk := X$
04 Return $sk, vk$

Proc. sgn$(sk, m)$
05 $r \xleftarrow{\$} \mathbb{Z}_p$
06 $R \leftarrow g^r$
07 $c \leftarrow H(R, m)$
08 $s \leftarrow r + xc$
09 $\sigma \leftarrow (c, s)$
10 Return $\sigma$

Proc. vfy$(vk, m, \sigma)$
11 $(c, s) \leftarrow \sigma$
12 $R \leftarrow g^s / X^c$
13 $\bar{c} \leftarrow H(R, m)$
14 $v \leftarrow \bar{c} =_? c$
15 Return $v$

Proc. $H(R, m)$
16 $\tilde{R} \leftarrow \varphi(R)$
17 $st_H \leftarrow new()$
19 $update\langle st_H \rangle(\tilde{R})$
21 $update\langle st_H \rangle(m)$
23 $h \leftarrow digest(st_H)$
24 $c \leftarrow \psi(h)$
25 Return $c$
```

Fig. 9. Left: Schnorr DSS. **Right:** Construction of H from an iterative hash function.

Practical hash functions don't natively implement the required $\mathbb{G} \times \mathcal{M} \to \mathbb{Z}_p$ mapping, but instead a mapping of the form $\tilde{H} \colon \{0,1\}^* \to \{0,1\}^l$ for some fixed l. To obtain a practical scheme from the algorithms of Fig. 9 (left) standards of Schnorr DSS also specify—at least implicitly— a fixed-length encoding $\varphi \colon \mathbb{G} \to \{0,1\}^L$ and a conversion function $\psi \colon \{0,1\}^l \to \mathbb{Z}_p$ and set([12]Some specifications (like BIP 340 [25]) generalize the right-hand side by computing $\psi(\tilde{H}(\text{aux}_1 \parallel \varphi(R) \parallel \text{aux}_2 \parallel m \parallel \text{aux}_3))$, where the (public) auxiliary inputs $\text{aux}_1, \text{aux}_2, \text{aux}_3$ may include a copy of the verification key, a cryptographic certificate on the key, or similar.)

$$H \colon \mathbb{G} \times \mathcal{M} \to \mathbb{Z}_p;\ (R, m) \mapsto \psi(\tilde{H}(\varphi(R) \parallel \text{m}))\ .^{12}$$

That is, using our hash function notation from Fig. 7 for $\tilde{H}$, H is implemented as in Fig. 9 (right).[12] In practice, roughly, for ECC based versions defined over

[12] Following Footnote 12, $update\langle st_H \rangle(\text{aux}_1)$ might have to be added to Fig. 9 in line 18, $update\langle st_H \rangle(\text{aux}_2)$ in line 20, and/or $update\langle st_H \rangle(\text{aux}_3)$ in line 22. Correspondingly, aux_1, aux_2 and aux_3 should be appended to sk and vk.

Weierstrass curves where group elements coincide with curve points R represented by two finite-field coordinates (R_x, R_y), function φ will output a bit serialization of either (R_x, R_y) or just R_x,[13] and function ψ will convert its l-bit input, or a truncation thereof, to a non-negative integer and output the latter reduced modulo p. The finite-field based versions are even simpler: ψ is as above, and φ does nothing beyond serializing the finite field element R to a bit-string.

3 Digital Signatures with Outsourced Hashing

As motivated in Sect. 1, the DSS with outsourced hashing (DSSwOH) notion formalizes the idea of distinguishing between the (VIL) hashing component and the (FIL) core signing components of a DSS. Here, the term hashing does not necessarily have to 1:1 correspond with invoking a cryptographic hash function like SHA2 or SHA3. Rather, as we shall see in the examples, it may make sense to let the core signing routine initiate and/or finalize a SHA computation. We start with formalizing the DSSwOH syntax illustrated in Fig. 4.

DSSwOH. A *digital signature scheme with outsourced hashing* **(DSSwOH)** for a message space $\mathcal{M}$ consists of a signing key space $\mathcal{SK}$, a verification key space $\mathcal{VK}$, a signature space $\mathcal{S}$, a hash seed space $\mathcal{HS}$, a hash value space $\mathcal{HV}$, a state space $\mathcal{ST}$, a key generation algorithm $\text{gen} \to \mathcal{SK} \times \mathcal{VK}$, a hash function $\mathit{hash}\colon \mathcal{HS} \times \mathcal{M} \to \mathcal{HV}$, and algorithms $\mathcal{SK} \to \text{sgn.ini} \to \mathcal{ST} \times \mathcal{HS}$ and $\mathcal{ST} \times \mathcal{HV} \to \text{sgn.fin} \to \mathcal{S}$ and $\mathcal{VK} \times \mathcal{S} \to \text{vfy.ini} \to \mathcal{ST} \times \mathcal{HS}$ and $\mathcal{ST} \times \mathcal{HV} \to \text{vfy.fin} \to \{0,1\}$. By composing the latter components in the way specified in Fig. 10 we obtain the associated signing algorithm $\mathcal{SK} \times \mathcal{M} \to \text{sgn} \to \mathcal{S}$ and verification algorithm $\mathcal{VK} \times \mathcal{M} \times \mathcal{S} \to \text{vfy} \to \{0,1\}$.

For correctness we expect the same of a DSSwOH as of a DSS, i.e., that for all $m \in \mathcal{M}$ after $(sk, vk) \leftarrow \text{gen}$ and $\sigma \leftarrow \text{sgn}(sk, m)$ and $v \leftarrow \text{vfy}(vk, m, \sigma)$ we have $v = 1$, where the sgn and vfy algorithms are those of Fig. 10. Similarly, as a first security notion we let $\mathbf{UF}^{\text{e,s}}_{\text{AA}} := \mathbf{UF}^{\text{e,s}}$ (with the latter referring to the games of Fig. 8). Note this notion assumes atomic execution of signing and verifying, and doesn't capture the DSSwOH peculiarities. We present corresponding refinements in Sects. 2 and 3.

```
Proc. sgn($sk, m$)                      Proc. vfy($vk, m, \sigma$)
00 $st, hs \leftarrow$ sgn.ini($sk$)     04 $st, hs \leftarrow$ vfy.ini($vk, \sigma$)
01 $hv \leftarrow hash(hs, m)$           05 $hv \leftarrow hash(hs, m)$
02 $\sigma \leftarrow$ sgn.fin($st, hv$) 06 $v \leftarrow$ vfy.fin($st, hv$)
03 Return $\sigma$                       07 Return $v$
```

Fig. 10. Algorithms sgn, vfy associated with a DSSwOH scheme. Note how the two algorithms correspond with the outer boxes of Fig. 4.

[13] Note that φ is 1:1 in the former case and 2:1 in the latter case. Precisely, in the 2:1 case we have $\varphi(R') = \varphi(R) \iff R' \in \{R, R^{-1}\}$.

To exercise our definition, we specify a DSSwOH version of the (Interactive HtS) Schnorr DSS, where we distribute the initial and final steps of the H specification of Fig. 9 (right) to the sgn.ini, vfy.ini, sgn.fin, vfy.fin algorithms. While technically there might be different ways to do this, the version we present below is the most natural one as: (a) the outsourced *hash* component does not 'have to get involved with number theory'; specifically: it neither has to encode group elements, serialize finite field elements, compress elliptic curve points, etc. as part of a φ implementation, nor does it have to implement modular reduction for the 'mod p' operation of a ψ implementation. And (b) leakage to a potential adversary is minimized, as a maximum of information is hashed before being provided to the outsourced party. In Sect. 5 we will prove that this DSSwOH is generally insecure, even if hashing is done honestly. In contrast, surprisingly, in Sect. 6 we prove that the security can be salvaged for very specific configurations of parameters, even if honest hashing cannot be assumed.

Starting with the Schnorr DSS with iterative hashing of Fig. 9 (left+right), in Fig. 11 we specify corresponding DSSwOH algorithms. Note that composing the DSSwOH algorithms as in Fig. 10 yields precisely the algorithms of the Schnorr DSS of Fig. 9.

```
**Proc.** gen
00 $x \xleftarrow{\$} \mathbb{Z}_p$
01 $X \leftarrow g^x$
02 $sk := x$
03 $vk := X$
04 Return $sk, vk$

**Proc.** $hash(hs, m)$
05 $update\langle st_H \rangle(m)$
07 $h \leftarrow digest(st_H)$
08 $hv := h$
09 Return $hv$
```

```
**Proc.** sgn.ini$(sk)$
10 $r \xleftarrow{\$} \mathbb{Z}_p$
11 $R \leftarrow g^r$
12 $\tilde{R} \leftarrow \varphi(R)$
13 $st_H \leftarrow new()$
15 $update\langle st_H \rangle(\tilde{R})$
17 $st := (x, r)$
18 $hs := st_H$
19 Return $st, hs$

**Proc.** sgn.fin$(st, hv)$
20 $c \leftarrow \psi(h)$
21 $s \leftarrow r + xc$
22 $\sigma := (c, s)$
23 Return $\sigma$
```

```
**Proc.** vfy.ini$(vk, \sigma)$
24 $R \leftarrow g^s / X^c$
25 $\tilde{R} \leftarrow \varphi(R)$
26 $st_H \leftarrow new()$
28 $update\langle st_H \rangle(\tilde{R})$
30 $st := c$
31 $hs := st_H$
32 Return $st, hs$

**Proc.** vfy.fin$(st, hv)$
33 $\bar{c} \leftarrow \psi(h)$
34 $v \leftarrow \bar{c} =_? c$
35 Return $v$
```

Fig. 11. Schnorr DSS with outsourced hashing. [17]([17] Extending Footnotes 12,13, $update\langle st_H \rangle(\text{aux}_1)$ might have to be added in lines 14,27, $update\langle st_H \rangle(\text{aux}_2)$ in lines 16,29, and/or $update\langle st_H \rangle(\text{aux}_3)$ in line 06. Correspondingly, aux_1, aux_2 and aux_3 should be appended to sk and vk in lines 02 and 03.)

4 DSSwOH Security with Benign Hashing

After introducing a formal syntax for DSS with outsourced hashing (DSSwOH) in Sect. 3, we move on to defining the security of this primitive. Akin to the differentiation between passive (CPA) and active (CCA) adversaries in the PKE world, our diverse DSSwOH security models differentiate between benign hashing and malicious hashing. When hashing is benign, the adversary can influence message inputs and observe the exchange of hash seeds hs and hash values hv in Fig. 4. When hashing is malicious, the adversary takes control of the hashing implementation and can return arbitrary hash values to the signing core. Sufficiency of the benign model could be claimed for cases where hashing is implemented on the same computer as the signing core, just at a different level of the software stack. However, in deployment scenarios involving trusted hardware, i.e., where applications outside of specifically protected computing cores are assumed non-trustworthy by default, relying on the benign model doesn't make sense and the malicious model has to be used. The current section is focused on the benign setting. Our treatment of the malicious setting is in Sect. 6.

Our security games for DSSwOH are based on those of regular DSS of Fig. 8 (and of [21]), with three major differences (two compared to [21]): (1) As a DSSwOH consists of more algorithms than a regular DSS, our games reflect this by adding the corresponding additional oracles, maintaining a 1:1 relationship between algorithms and oracles; (2) As signing and verifying in a DSSwOH is stateful, and multiple messages could be signed concurrently, this has to be reflected in our games by introducing a session concept. Without the support of concurrent sessions, attacks wouldn't be covered where an adversary opens multiple sessions in parallel and lets them interact with each other in a malicious way. (Our attack on the Schnorr DSSwOH depends precisely on this capability.) We implement sessions via in-game session identifiers that the adversary can pick arbitrarily. (3) While we define existential and strong unforgeability notions (UF) for DSSwOH as we did for DSS, we also import two more technical notions from [21] (HUF) that are based on the hashes of messages rather than on the messages themselves. In a nutshell, an adversary wins the HUF game for DSSwOH if it comes up with a valid fresh hash-seed/hash-value pair in the weaker case (existential unforgeability) and a valid fresh hash-seed/hash-value/signature triplet in the stronger case (strong unforgeability).

We specify our $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}, \mathbf{UF}^{\mathrm{s}}_{\mathrm{BB}}, \mathbf{HUF}^{\mathrm{e}}_{\mathrm{BB}}, \mathbf{HUF}^{\mathrm{s}}_{\mathrm{BB}}$ games in Fig. 12, where the BB subindex ($\underline{\text{b}}$enign-$\underline{\text{b}}$enign) indicates that the hashing in both signing and verifying is required to be benign. The benign behavior is technically enforced by lines 14,25 of Fig. 12 that ensure that any hash value provided by the adversary to the SgnFin and VfyFin oracles was indeed first generated by hashing a message in line 12 or 23, respectively.

Our games use session identifiers sid to distinguish sessions. The session management is not explicit in Fig. 12, but indicated by the curly top-down arrows. Precisely, the session management guarantees that (1) for each session identifier sid, the oracles are strictly called in the indicated order, i.e., SgnIni $\rightarrow$ Hash $\rightarrow$ SgnFin and VfyIni $\rightarrow$ Hash $\rightarrow$ VfyFin; this rule could explicitly be enforced with additional code that implements a state machine; (2) for each session identifier sid, variables st, hs, hv, m are state-persistent; that is: hs in line 12 is the one of lines 10,11 for the same sid; and hv in lines 14,15 is the one of lines 12,13 for the same *sid*; this could be explicitly implemented with additional array variables that map session identifiers to variables. We use this compact game notation as it cleanly puts the focus on what really matters, namely the very spirit of the security notions. Readers who prefer a fully explicit presentation are referred to Appendix A and Fig. 17, which are equivalent but explicit.

```
INITIALIZATIONS: $\mathrm{A}, \mathrm{A}^{\cdot}, \mathrm{B}, \mathrm{B}^{\cdot}, \mathrm{V}, \mathrm{V}^{\cdot}, \mathrm{W}, \mathrm{W}^{\cdot} \leftarrow \emptyset$

Game $\mathbf{UF}^{\mathrm{e,s}}_{\mathrm{BB}}(\mathcal{A})$
00 $sk, vk \leftarrow \mathrm{gen}$
01 $\mathcal{A}(vk)$
02^e Promise $\mathrm{V} \subseteq \mathrm{A}$
03^s Promise $\mathrm{V}^{\cdot} \subseteq \mathrm{A}^{\cdot}$
04 Lose

Game $\mathbf{HUF}^{\mathrm{e,s}}_{\mathrm{BB}}(\mathcal{A})$
05 $sk, vk \leftarrow \mathrm{gen}$
06 $\mathcal{A}(vk)$
07^e Promise $\mathrm{W} \subseteq \mathrm{B}$
08^s Promise $\mathrm{W}^{\cdot} \subseteq \mathrm{B}^{\cdot}$
09 Lose

Oracle SgnIni($sid$)
10 $st, hs \leftarrow \mathrm{sgn.ini}(sk)$
11 Share $hs$

Oracle SgnHash($sid\colon m$)
12 $hv \leftarrow hash(hs, m)$
13 Share $hv$

Oracle SgnFin($sid\colon \overline{hv}$)
14 Require $\overline{hv} = hv$
15 $\sigma \leftarrow \mathrm{sgn.fin}(st, hv)$
16 Share $\sigma$
17 $\mathrm{A} \xleftarrow{\cup} \{m\}$
18 $\mathrm{A}^{\cdot} \xleftarrow{\cup} \{(m, \sigma)\}$
19 $\mathrm{B} \xleftarrow{\cup} \{(hs, hv)\}$
20 $\mathrm{B}^{\cdot} \xleftarrow{\cup} \{(hs, hv, \sigma)\}$

Oracle VfyIni($sid\colon \sigma$)
21 $st, hs \leftarrow \mathrm{vfy.ini}(vk, \sigma)$
22 Share $hs$

Oracle VfyHash($sid\colon m$)
23 $hv \leftarrow hash(hs, m)$
24 Share $hv$

Oracle VfyFin($sid\colon \overline{hv}$)
25 Require $\overline{hv} = hv$
26 $v \leftarrow \mathrm{vfy.fin}(st, hv)$
27 Share $v$
28 If $v$: $\mathrm{V} \xleftarrow{\cup} \{m\}$
29 If $v$: $\mathrm{V}^{\cdot} \xleftarrow{\cup} \{(m, \sigma)\}$
30 If $v$: $\mathrm{W} \xleftarrow{\cup} \{(hs, hv)\}$
31 If $v$: $\mathrm{W}^{\cdot} \xleftarrow{\cup} \{(hs, hv, \sigma)\}$
```

Fig. 12. DSSwOH security games for benign hashing. Line 02 is part of $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}$ but not of $\mathbf{UF}^{\mathrm{s}}_{\mathrm{BB}}$, while line 03 is part of $\mathbf{UF}^{\mathrm{s}}_{\mathrm{BB}}$ but not of $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}$, and similarly lines 07,08 for games $\mathbf{HUF}^{\mathrm{e}}_{\mathrm{BB}}, \mathbf{HUF}^{\mathrm{s}}_{\mathrm{BB}}$. The top-down arrows represent the session management (see text, and the explicit version in Fig. 17). The semantics of the set variables $\mathrm{A}, \mathrm{V}, \mathrm{A}^{\cdot}, \mathrm{V}^{\cdot}$ is as in Fig. 8. The semantics of the set variables $\mathrm{B}, \mathrm{W}, \mathrm{B}^{\cdot}, \mathrm{W}^{\cdot}$ is corresponding, but for hash values instead of messages.

Definition 2 (Unforgeability of DSSwOH, BB case). *Consider the* $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}, \mathbf{UF}^{\mathrm{s}}_{\mathrm{BB}}, \mathbf{HUF}^{\mathrm{e}}_{\mathrm{BB}}, \mathbf{HUF}^{\mathrm{s}}_{\mathrm{BB}}$ *games of Fig. 12. We define unforgeability advantages of an adversary* $\mathcal{A}$ *as per* $\mathbf{Adv}^{\text{uf-e-bb}}(\mathcal{A}) := \Pr[\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}(\mathcal{A})]$, $\mathbf{Adv}^{\text{uf-s-bb}}(\mathcal{A}) := \Pr[\mathbf{UF}^{\mathrm{s}}_{\mathrm{BB}}(\mathcal{A})]$, $\mathbf{Adv}^{\text{huf-e-bb}}(\mathcal{A}) := \Pr[\mathbf{HUF}^{\mathrm{e}}_{\mathrm{BB}}(\mathcal{A})]$ *and* $\mathbf{Adv}^{\text{huf-s-bb}}(\mathcal{A}) := \Pr[\mathbf{HUF}^{\mathrm{s}}_{\mathrm{BB}}(\mathcal{A})]$, *respectively. Intuitively, we say that a DSSwOH is existentially (resp., strongly) unforgeable if* $\mathbf{Adv}^{\text{uf-e-bb}}(\mathcal{A})$ *(resp.,* $\mathbf{Adv}^{\text{uf-s-bb}}(\mathcal{A})$*) is negligible for all realistic* $\mathcal{A}$.

The $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}, \mathbf{UF}^{\mathrm{s}}_{\mathrm{BB}}$ games are in clear correspondence with the DSS games of Fig. 8, just that they replace the atomic operations with the outsourced ones. The $\mathbf{HUF}^{\mathrm{e}}_{\mathrm{BB}}, \mathbf{HUF}^{\mathrm{s}}_{\mathrm{BB}}$ notions are new for DSSwOH, and make the separation of hashing from the core signing more explicit. Concretely, where in $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}$ a forgery requires a valid signature on a fresh message, in $\mathbf{HUF}^{\mathrm{e}}_{\mathrm{BB}}$ a forgery requires a valid signature on a fresh hash-seed/hash-value pair. The conceptual difference is that if the adversary finds m, m' that hash with the same hash seed to the same hash value, i.e., if the problem is solely with the hash function, then this counts as a forgery in $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}$ but not in $\mathbf{HUF}^{\mathrm{e}}_{\mathrm{BB}}$. That is, $\mathbf{HUF}^{\mathrm{e}}_{\mathrm{BB}}$ captures precisely what it means to forge from the point of view of a signing core.

Unforgeability of DSSwOH, AB and BA cases. It is meaningful to also formalize mixed-case security definitions, i.e., where signing is atomic and verification uses benign hashing, and vice versa. Our game notations are such that deriving the mixed-case variants (AB and BA) is immediate by just cherry-picking the right oracle definitions from Figs. 8 and 12. For instance, game $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BA}}$ is like game $\mathbf{UF}^{\mathrm{e}}_{\mathrm{AA}} = \mathbf{UF}^{\mathrm{e}}$ (see Fig. 8 and Sect. 3) but with the signing oracle replaced with the middle column of Fig. 12.

5 Universal Forgeries for Schnorr DSSwOH with Benign Hashing

We show negative results for Schnorr signatures in the benign hashing setting (for positive results see Sect. 6.1). We demonstrate that many standardized versions of the Schnorr DSS become forgeable when operated with outsourced benign iterated hashing. Concretely, our attack consists of opening multiple concurrent signing sessions and letting them interact in a particular way so that attackers can craft universal forgeries, i.e., valid signatures on arbitrary messages. A key condition for this attack to succeed, is to extract sufficiently much information from the hash seed communicated to the outsourced hash implementation. As we will see, for many configurations of the Schnorr DSS this condition is fulfilled.

5.1 Attack on Concurrent Schnorr ZK-PoK

We start with making a closely related observation on the interactive Schnorr ZK-PoK proof system (see Sect. 2.4). Roughly, if the latter is implemented over an n-bit group (i.e., of order about 2^n), an adversary engaging with a prover in $l \geq n$ concurrent sessions can output, in addition to the l authentic hence valid proof transcripts $\{(R_i, c_i, s_i)\}_{1 \leq i \leq l}$, one additional non-authentic yet valid proof transcript (R^*, c^*, s^*). What differentiates this from just generating the additional transcript with the zero-knowledge simulator, is that the components R^*, c^*, s^* emerge in the same order as in regular protocol transcripts. Precisely: The adversary first outputs R^*, then fixes an arbitrary c^*, then completes the transcript by outputting s^*.[14]

The core of our attack is heavily inspired by recent work [11] on the (un-)forgeability of interactive versions of the Schnorr DSS, most prominently of blind Schnorr signatures. The authors of [11] propose a solver for the so-called ROS

[14] A simulator for the Schnorr protocol would typically first fix c^* and s^*, and then derive $R^* = g^{s^*}/X^{c^*}$ from them. This deviates from the order in which the values emerge in regular protocol runs.

problem, and show how this solver can be transformed into an effective attacker against the signature schemes. The ROS problem itself combines linear algebra with a random oracle: The challenge is to solve a system of linear equations some coefficients of which are derived via a random oracle from the very solution.[15] While the core of our attack reads almost the same as the one of [11], we feel that the ROS formalization of [11] is not general enough to imply our result, for at least two reasons: (1) While the ROS problem revolves around a random oracle, there is no random oracle in the Schnorr ZK-PoK system; and (2) while a crucial attack component in [11] is that the adversary modifies transcript components R_i by exposing them to multiplicative blinding, our attack on Schnorr signatures leaves these components intact. (In fact it *has* to leave them intact: Neither does our security model tolerate active adversaries, nor does the Schnorr ZK-PoK protocol provide an opportunity for applying blinding.)

Our attacker A against Schnorr's interactive ZK-PoK protocol is specified in Fig. 13, together with its subroutines $\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3$. We explain the details in the following. Lines 00–03 open l concurrent sessions with the prover, receive the corresponding commitments R_i, and propose arbitrary two challenge values c_i^0, c_i^1 for each of them. (At this point in time the latter are just selected; they are not yet sent.) In line 04, subroutine A_1 derives from its R_i and c_i^0, c_i^1 inputs the commitment R^* that will be part of the forged transcript. The adversary, in line 05, then picks an arbitrary challenge c^* for this commitment. This value is forwarded to subroutine A_2 which continues the computation of A_1, this time outputting a bit vector $b_1, \ldots, b_l$ that indicates for each session which of the two earlier-proposed challenge values c_i^0, c_i^1 shall be sent to the prover. (In line 08, the symbol $\bar{c}_i$ is introduced to explicitly indicate the selected challenge.) Lines 07–11 complete the sessions, and record their transcripts in variables $\tau_1, \ldots, \tau_l$. The sessions' response values s_i are, in line 12, also provided to subroutine A_3 so that the latter can contribute the response s^* that completes the forged transcript τ^*. Because the used attack techniques overlap with those of [11] we defer the explanation of subroutines $\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3$, and the proof that transcript τ^* is always valid, to Appendix B.[16]

Lemma 1. *The attack in Fig. 13 derives a fresh Schnorr ZK proof transcript* (R^*, c^*, s^*) *in the order* $R^* \to c^* \to s^*$ *by interacting with* $l \geq n$ *concurrently executed transcript generators.*

5.2 Attack on Schnorr DSSwOH with Benign Hashing

We lift the attack algorithm A of Sect. 5.1 to an attack algorithm $\bar{\mathrm{A}}$ against Schnorr DSSwOH in the $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}$ model of Sect. 4 and the $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BA}}$. The core computational steps of $\bar{\mathrm{A}}$ and A are the same; what differs is how $\bar{\mathrm{A}}$ derives the values R_i, s_i and fixes the values c_i^b, c^*, i.e., how it implements the steps that

[15] For a precise formulation see Definition 4 in Appendix B.

[16] There are no such subroutines in [11]. While the attack techniques overlap, the abstractions used in [11] and by us are disjoint.

```
Proc. A($X$)
00 For $i \in [1..l]$:
01     Open proof session $\#i$
02     In session $\#i$: Receive $R_i$
03     Pick any distinct $c_i^0, c_i^1$
04 $st, R^* \leftarrow \mathrm{A}_1(R_1..R_l, c_1^0..c_l^0, c_1^1..c_l^1)$
05 Pick any $c^*$
06 $st, b_1..b_l \leftarrow \mathrm{A}_2(st, c^*)$
07 For $i \in [1..l]$:
08     $\bar{c}_i := c_i^{b_i}$
09     In session $\#i$: Send $\bar{c}_i$
10     In session $\#i$: Receive $s_i$
11     Let $\tau_i := (R_i, \bar{c}_i, s_i)$
12 $s^* \leftarrow \mathrm{A}_3(st, s_1..s_l)$
13 Let $\tau^* := (R^*, c^*, s^*)$
14 Return $\tau_1 \ldots \tau_l, \tau^*$

Proc. A$_1$($R_1..R_l, c_1^0..c_l^0, c_1^1..c_l^1$)
15 For $i \in [1..l]$:
16     Let $\lambda_i : x_1..x_l \mapsto (x_i - c_i^0)/(c_i^1 - c_i^0)$
17 Let $P: x_1..x_l \mapsto \sum_{i=1}^{l} w_i \lambda_i(x_1..x_l)$
18 Find $\alpha_0..\alpha_l$ s.t. $P: x_1..x_l \mapsto \alpha_0 + \sum_{i=1}^{l} \alpha_i x_i$
19 $R^* \leftarrow R_1^{\alpha_1} \cdots R_l^{\alpha_l}$
20 $st := \alpha_0..\alpha_l$
21 Return $st, R^*$

Proc. A$_2$($st, c^*$)
22 Find $b_1..b_l$ s.t. $\sum_{i=1}^{l} w_i b_i = \alpha_0 + c^*$
23 Return $st, b_1..b_l$

Proc. A$_3$($st, s_1..s_l$)
24 $s^* \leftarrow \alpha_1 s_1 + \ldots + \alpha_l s_l$
25 Return $s^*$
```

Fig. 13. Attack on Schnorr ZK-PoK. Weights $w_1, \ldots, w_l$ of line 17 are such that the subset-sum problem of line 22 can be efficiently solved. A natural choice is requiring $l = n$ and setting $w_i = 2^{l-i}$; in this case line 22 reduces to computing a binary expansion. See [11] for a discussion of other possible choices (that also might get along with $l < n$).

remained unspecified on the left hand side of Fig. 13. The full attack specification in Fig. 14 shows how $\bar{\mathrm{A}}$ extracts R_i, s_i from sgn.ini and sgn.fin queries, respectively. $\bar{\mathrm{A}}$ then chooses values c_i^b, c^* in the $c \leftarrow H(R, m)$ spirit of line 07 of Fig. 9, where the underlying messages m_i^b are arbitrary but distinct, and m^* is the target message to (universally) forge on. By Lemma 1 this always leads to a valid ZK transcript $\tau^* = (R^*, c^*, s^*)$ from which the forgery $\sigma^* = (s^*, c^*)$ on m^* is readily extracted.

A technical precondition for this attack to succeed is the ability to extract the ZK commitment R_i from the hash seed hs_i in line 03. (Recall the extraction principle from Sect. 2.2.) Whether this is possible or not depends on implementational details.

The following combines the above discussion with the results of Lemma 1.

Theorem 1. *Consider a Schnorr DSSwOH implementation as of Fig. 11 over a group $\mathbb{G}$ and assume the ZK commitment R can always be extracted from the hash seeds hs output by the* sgn.ini *algorithm. Then the attack of Fig. 14, when engaging with $l \geq \lfloor \log_2 |\mathbb{G}| \rfloor$ concurrent sessions in the* $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}$ *game, always comes up with a valid (universal) forgery.*

On the 🔴 and 🟧 marks of Fig. 6. The previous theorem lays the basis on which we compute part of Fig. 6, namely, the 🔴 and the 🟧 marks. In this figure, we study five variants of Schnorr signature schemes (SDSA [5], ECSDSA [4],

```
Proc. $\bar{\mathrm{A}}(vk)$
Pick distinct $sid_1 \mathinner{..} sid_l$
For $i \in [1 \mathinner{..} l]$:
    $hs_i \leftarrow \mathrm{SgnIni}(sid_i)$
    Extract $R_i$ from $hs_i$
    Pick distinct $m_i^0, m_i^1$
    $c_i^0 \leftarrow hash(hs_i, m_i^0)$; $c_i^1 \leftarrow hash(hs_i, m_i^1)$
    If $c_i^0 = c_i^1$: Abort
$st, R^* \leftarrow \mathrm{A}_1(R_1 \mathinner{..} R_l, c_1^0 \mathinner{..} c_l^0, c_1^1 \mathinner{..} c_l^1)$
Pick fresh $m^*$
$c^* \leftarrow H(R^*, m^*)$
$st, b_1 \mathinner{..} b_l \leftarrow \mathrm{A}_2(st, c^*)$
For $i \in [1 \mathinner{..} l]$:
    Let $\bar{m}_i := m_i^{b_i}$ and $\bar{c}_i := c_i^{b_i}$
    $hv_i \leftarrow \mathrm{SgnHash}(sid_i \colon \bar{m}_i)$
    $\sigma_i \leftarrow \mathrm{SgnFin}(sid_i \colon hv_i)$
    $s_i, c_i \leftarrow \sigma_i$ // $c_i = \bar{c}_i$
$s^* \leftarrow \mathrm{A}_3(st, s_1 \mathinner{..} s_l)$
$\sigma^* := (s^*, c^*)$
```

Fig. 14. Lifting of Fig. 13 to an attack against Schnorr DSSwOH in the $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}$ security model. The subroutines $\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3$ are as in Fig. 13. Formally, delivering the forgery (m^*, σ^*) entails one extra $\mathrm{VfyIni}(\sigma^*) \rightarrow \mathrm{VfyHash}(m^*) \rightarrow \mathrm{VfyFin}(hv^*)$ cycle.

ECSDSA Optimized [5], BIP 340 [25] (see Footnote 8), and ECFSDSA [5])[17] when the hashing is outsourced as in Fig. 11.

We look into 252 different configurations (a.k.a. instantiations) in total: each column specifies a size of the finite field over which the DSSwOH scheme is defined and each row specifies which hash function is included in the scheme. We use four colors to label the 'security levels' of these instantiations. We clarify what we mean by the security levels throughout the paper and dedicate this section only to the levels represented with 🔴 and 🟧 marks. As an example, the upper-left mark of ECSDSA (b) reads as: "The security level of the outsourced ECSDSA defined over a 160-bit finite field and instantiated with SHA1 is represented with 🔴."

The 🔴 mark symbolizes a specific instantiation that is broken in the $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}$ security model. Breaking such instantiation is an artifact of Theorem 1 which assumes that the commitment R is extractable from the hash seed hs. Three key factors essential for the extractability of R from $hs := (cv, B, cnt)$ are: (1) the invertibility of φ, (2) the aggregated bit-length of the *update* inputs in the sgn.ini algorithm (i.e., $|\varphi(R_i)|$[18]), and the blocksize ρ of the hash function.

With the correct configurations, one way to extract R is to invert φ and apply φ^{-1} over B's content. Some instantiations marked with 🔴 define φ as an encoding function that serializes its input in an injective manner[19] and allow L (the output size of φ) to be smaller than the rate ρ. The latter condition means that the procedure *update* stores $\varphi(R)$ in the buffer B. The former condition implies that it is possible to recover R by inverting $\varphi(R)$. Together, these conditions allow the extraction of R from hs.

[17] In a nutshell, the EC-based variants differ from one another with the definition of the encoding function φ and with the auxiliary inputs included in the hashing. For instance, they all define $\mathrm{aux}_1, \mathrm{aux}_2, \mathrm{aux}_3$ to be resp. $\epsilon, \epsilon, \epsilon$ except for BIP 340 which defines aux_2 as vk_{x}, the x coordinate of the public key vk.

[18] Extending Footnotes 12, 13, the adapted criterion would be $|\mathrm{aux}_1 \mathbin{\|} \varphi(R_i) \mathbin{\|} \mathrm{aux}_2|$.

[19] For instance, φ maps an EC point to the bitstring representation of its coordinates.

Another method to extract R from hs is by inverting the hashing steps leading to cv. Looking into sponge-based hash functions, we notice that if the chaining value is an outcome of a single evaluation of the underlying permutation, then it is possible to recover the input string on which the hash function has been evaluated. More specifically, having $cv = \Pi(\mathrm{IV} \oplus X_1 \mathbin{\|} 0^\kappa)$ where Π is a public permutation, X_1 is a private string and IV is a public initialization vector, it is possible recover X_1 by computing $X_1 \mathbin{\|} 0^\kappa = \Pi^{-1}(cv) \oplus \mathrm{IV}$. This case actually occurs (in 13 instantiations of Fig. 6 where SHA3 is used) when $L \in \rrbracket \rho, 2\rho \llbracket$ (correspondingly $|\mathrm{aux}_1| + L + |\mathrm{aux}_2|$ is in this set). Thus the recovered X_1 would be the ρ prefix bits of $\varphi(R)$ and the value stored in B would correspond to the $L - \rho$ suffix bits of $\varphi(R)$. Now, that $\varphi(R)$ is restored, it is possible to extract R if φ is invertible. We refer the reader to the full version for a visual representation of sponge constructions.

As for the 🟧 marks, Theorem 1 does not apply straightforwardly: In these instantiations, φ is defined as a 2:1 function mapping an elliptic curve point to the bitstring describing its x coordinate. This means that it is possible to extract R_x of hs but not R_y and thus not the entire element R. Guessing the y coordinate sums up to recovering two different y coordinates using the elliptic curve equation, then guessing a single bit that determines which of the two is used. If this bit is known to the attacker (e.g. assuming all the managed EC points are "positive"), then extracting R of hs would be possible and the instantiation would be broken in the $\mathbf{UF}^{\mathrm{e}}_{\mathrm{BB}}$ security model by Theorem 1.

6 DSSwOH Security with Malicious Hashing

The security definitions of Sect. 4 assumed that the message hashing of Fig. 4 is executed honestly. (The adversary controls the messages and observes the resulting hash seeds/values, but the hash values are always properly computed.) In this section, we define models that drop this condition and assume dishonest hashing, i.e., assume the message hashing is performed by the adversary. As explained in Sect. 4 these security models are indispensable in environments involving trusted hardware.

Lines 14,25 of $\mathbf{UF}^{\mathrm{e,s}}_{\mathrm{BB}}$ (Fig. 12) ensure that there exists an honest hashing operation for every hash value provided by the adversary. However, removing these lines is insufficient to convert a benign hashing setting into a malicious one. Indeed, in the malicious hashing setting, the recorded messages in lines 17,18,28,29 do not necessarily exist or might not be communicated by the adversary. We hence resort to defining only the hash value centric HUF notions in the malicious hashing setting.

We specify $\mathbf{HUF}^{\mathrm{e}}_{\mathrm{MM}}, \mathbf{HUF}^{\mathrm{s}}_{\mathrm{MM}}$ games in Fig. 15, where the MM subindices ($\underline{\mathrm{m}}$alicious-$\underline{\mathrm{m}}$alicious) indicate that the hashing in both signing and verifying is allowed to be malicious.

Definition 3 (Unforgeability of DSSwOH, MM case). *Consider the* $\mathbf{HUF}^{\mathrm{e}}_{\mathrm{MM}}, \mathbf{HUF}^{\mathrm{s}}_{\mathrm{MM}}$ *games of Fig. 15. We define unforgeability advantages of an* *adversary $\mathcal{A}$ as per* $\mathbf{Adv}^{\text{huf-e-mm}}(\mathcal{A}) \coloneqq \Pr[\mathbf{HUF}^{\mathrm{e}}_{\mathrm{MM}}(\mathcal{A})]$ *and* $\mathbf{Adv}^{\text{huf-s-mm}}(\mathcal{A}) \coloneqq \Pr[\mathbf{HUF}^{\mathrm{s}}_{\mathrm{MM}}(\mathcal{A})]$*, respectively. Intuitively, we say that a DSSwOH is existentially (respectively, strongly) unforgeable in the malicious-malicious setting if* $\mathbf{Adv}^{\text{huf-e-mm}}(\mathcal{A})$ *(resp.,* $\mathbf{Adv}^{\text{huf-s-mm}}(\mathcal{A})$*) is negligible for all realistic $\mathcal{A}$.*

```
INITIALIZATIONS: $\mathrm{B}, \mathrm{B}^{\cdot}, \mathrm{W}, \mathrm{W}^{\cdot} \leftarrow \emptyset$

Game $\mathbf{HUF}^{\mathrm{e,s}}_{\mathrm{MM}}(\mathcal{A})$
00    $sk, vk \leftarrow$ gen
01    $\mathcal{A}(vk)$
02^e  Promise $\mathrm{W} \subseteq \mathrm{B}$
03^s  Promise $\mathrm{W}^{\cdot} \subseteq \mathrm{B}^{\cdot}$
04    Lose

Oracle SgnIni($sid$)
05    $st, hs \leftarrow$ sgn.ini($sk$)
06    Share $hs$

Oracle SgnFin($sid$: $hv$)
07    $\sigma \leftarrow$ sgn.fin($st, hv$)
08    Share $\sigma$
09    $\mathrm{B} \overset{\cup}{\leftarrow} \{(hs, hv)\}$
10    $\mathrm{B}^{\cdot} \overset{\cup}{\leftarrow} \{(hs, hv, \sigma)\}$

Oracle VfyIni($sid$: $\sigma$)
11    $st, hs \leftarrow$ vfy.ini($vk, \sigma$)
12    Share $hs$

Oracle VfyFin($sid$: $hv$)
13    $v \leftarrow$ vfy.fin($st, hv$)
14    Share $v$
15    If $v$: $\mathrm{W} \overset{\cup}{\leftarrow} \{(hs, hv)\}$
16    If $v$: $\mathrm{W}^{\cdot} \overset{\cup}{\leftarrow} \{(hs, hv, \sigma)\}$
```

Fig. 15. DSSwOH security games for malicious hashing. See Fig. 12 for further explanations of the notation.

Security notions where one party is benign and the other is malicious can be straightforwardly defined by considering hybrids of Figs. 12 and 15. In continuation of the heading after Definition 2 we formalize this in the following.

UNFORGEABILITY OF DSSWOH, MB AND BM CASES. Define games $\mathbf{HUF}^{\mathrm{e}}_{\mathrm{BM}}$, $\mathbf{HUF}^{\mathrm{s}}_{\mathrm{BM}}$ by combining the middle column of Fig. 12 with the right column of Fig. 15, and games $\mathbf{HUF}^{\mathrm{e}}_{\mathrm{MB}}, \mathbf{HUF}^{\mathrm{s}}_{\mathrm{MB}}$ by combining the middle column of Fig. 15 with the right column of Fig. 12. Also define corresponding symbols for the advantage terms, following the pattern of Definitions 2 and 3.

Remark 1. Schnorr signature schemes can be trivially broken when the outsourced hashing on the verifier's side is malicious. The attacker computes a valid hash value and corresponding signature without querying the signing oracle. One interpretation of this attack is that security in the face of malicious hashing of verification cannot be reached. Another interpretation is that our $\mathbf{HUF}^{\mathrm{e,s}}_{\mathrm{AM}}$, $\mathbf{HUF}^{\mathrm{e,s}}_{\mathrm{BM}}$ and $\mathbf{HUF}^{\mathrm{e,s}}_{\mathrm{MM}}$ models are unnecessarily strong. Indeed, while in the secure hardware deployment scenarios it makes sense to assume malicious hashing on the signer side, the same does not have to hold on the verifier side. It is hence meaningful to focus on $\mathbf{HUF}^{\mathrm{e,s}}_{\mathrm{MB}}$ notions (matching the secure hardware case).

6.1 Security of Schnorr DSSwOH in the MB Case

We prove positive results for many Schnorr DSSwOH configurations (of Fig. 11) in the $\mathbf{HUF}^{\mathrm{e,s}}_{\mathrm{MB}}$ setting. These results do not contradict the negative findings of Sect. 5 as the latter only apply if R is extractable from the hash seed hs, while here we consider configurations where the contrary holds. Specifically, our positive results cover all the Schnorr DSSwOH instantiations where $|\varphi(R)|$ (correspondingly, $|\mathrm{aux}_1 \shortparallel \varphi(R) \shortparallel \mathrm{aux}_2|$) is a multiple of the rate ρ, and where the iterative hash function is a Merkle–Damgård (MD) based construction.[20] These instantiations are marked with ⬢ in Fig. 6.

Intuitively, the above conditions ensure that the inputs hashed in the signature initialization phase are well-aligned with MD block boundaries: In the terms of Fig. 7, after hashing the $\varphi(R)$ prefix, the buffer B in the hash seed hs is empty and the counter cnt in hs is a public constant (aux_1 and aux_2 are public strings, and the output size of φ is the constant L). Further, the input is properly processed by the hash function and fully compressed into the chaining value cv of hs.

As B and cnt are simulatable (public constants), then proving the security of the previous instantiations boils down to showing that the compressed chaining value cv contains no information about the input. Going back to Fig. 11, we notice that the sequence of steps that allow the computation of cv are equivalent to running h_1 of Fig. 16 (left) and that the steps that allow the computation of a digest can be regrouped into h_2 of Fig. 16 (left). We then notice that if the alignment property holds, then h_1 is equivalent to a fixed length MD chain Fig. 16 (right) which, in its turn, can be modeled as a random oracle. (Lemma 2 states that the fixed-length MD construction is indifferentiable from a random oracle when the underlying compression function is modeled as ideal.[21]) Having h_1 be modeled as a random oracle means that cv leaks no information about the input. We give a short version of our result in Theorem 2 along with a simple proof, and a detailed version in the full version.

```
Proc. $h_1(\tilde{R})$
00 $st_H \leftarrow new()$
02 $update\langle st_H\rangle(\tilde{R})$
04 $(cv, \_, \_) \leftarrow st_H$
05 Return $cv$

Proc. $h_2(L; cv, m)$
06 $B \leftarrow \epsilon$; $cnt \leftarrow L$
07 $st_H \leftarrow (cv, B, cnt)$
08 $update\langle st_H\rangle(m)$
10 $h \leftarrow digest(st_H)$
11 Return $h$

Proc. $h_1(\tilde{R})$
12 $cv \leftarrow \mathrm{IV}$
13 $\tilde{R}_1 \shortparallel \ldots \shortparallel \tilde{R}_l \leftarrow \tilde{R}$
14 For $i \leftarrow 1$ to $l$:
15     $cv \leftarrow \mathrm{CF}(cv, \tilde{R}_i)$
16 Return $cv$
```

Fig. 16. Functions h_1, h_2 used in Theorem 2. h_1 (left) regroups lines 13–16 of Fig. 11 and h_2 regroups lines 05–07. The missing lines refer in this figure are referred to by those in footnote 17. It is helpful to think of $\tilde{R}$ as the encoded group element $\varphi(R)$, and L as the output length of φ (see Sect. 2.4). The right hand side version of this figure is a rewriting of h_1 as a fixed length MD construction.

[20] Our proofs can be extended to sponge-based constructions where $|\varphi(R)| \geq 2\rho$, however, none of the real-world instantiations in Fig. 6 meets this condition.

[21] Indifferentiability results on similar MD variants have been shown in [12] and [10] etc. However, to our knowledge, no positive result for the exact variant we use has been shown in literature.

Lemma 2. *Let* CF *be an ideal compression function for a fixed length MD construction. Modeling* CF *as ideal, we have that the fixed length MD construction is indifferentiable from a random oracle.*

Proof. In short, we construct a simulator that outputs randomly picked chaining values when queried. The simulator monitors the chain length and outputs the random oracle evaluation on the chained message input only when the chain is completed. The details of the proof and the simulator construction can be found in the full version.

Theorem 2. *Let S be the Schnorr DSS with outsourced hashing described in Fig. 11. Assuming that the discrete logarithm problem is hard and that finding collisions in $\psi \circ h_2$ (defined in Fig. 16) is hard, then modeling h_1 (of Fig. 16) as a random oracle implies that S meets $\mathbf{HUF}_{\mathrm{MB}}^{\mathrm{e,s}}$ security.*

Proof. In brief, we build a reduction that receives a group element and invokes our adversary on this same element (as a public key). It then harvests the output forgery, rewinds the adversary to the forgery point then invokes it with different random oracle outputs. If the forgeries are valid, then the reduction extracts the discrete logarithm correspondent to the group element. The success probability of this rewinding process is covered by the Forking Lemma. Some specific cases occur where the reduction is capable to extract the secret key with a single adversarial forgery and without rewinding: this happens when the provided forgery is correlated to a simulated signature. Additionally, the analysis takes care of the collisions between values computed by $\psi \circ h_2$ (when rewinding) by requiring that the composition meets a mild collision resistance security notion (see full version). The proof details can be found in the full version.

Lemma 2 and Theorem 2 both yield the following corollary:

Corollary 1. *Let S be defined in Fig. 11. If the inputs to the hashing steps in the signature initialization are well aligned with the hash function block size, then S is unforgeable in the $\mathbf{HUF}_{\mathrm{MB}}^{\mathrm{e,s}}$ sense.*

On the ⬢ marks of Fig. 6. Back to Fig. 6, the ⬢ marks refer to instantiations that are unforgeable in the $\mathbf{HUF}_{\mathrm{MB}}^{\mathrm{e,s}}$ sense. The previous corollary helps determine whether an instantiation should be marked with ⬢: if $|\mathrm{aux}_1|+L+|\mathrm{aux}_2| = 0 \bmod \rho$ then the instantiation is secure (see Sect. 2.4 for the definitions of L and ρ). Many instantiations immediately fit into this criterion: the hash function defines ρ, the scheme defines aux_1 and aux_2, and L can be decided by looking into.

Concerning BIP 340 (see Footnote 8), notable configurations emerge when the underlying field is ECC 384 and the hash function is either SHA1, SHA2-224 or SHA2-256. Recalling footnote 21, BIP 340 defines $(\mathrm{aux}_1, \mathrm{aux}_2)$ as (ϵ, vk_x) and φ as the mapping from an EC point input to its x coordinate. This implies that when the field size is 384 bits long, $|\mathrm{aux}_1| + L + |\mathrm{aux}_2| = 768$ bits. Because $\rho = 512$ for those hash functions, the first 512 input-bits are compressed into the chaining value and the last 256 input-bits are stored in the buffer of the hash seed. Since the latter 256 bits correspond to the suffix of the public value $\mathrm{aux}_2 = vk_{\mathrm{x}}$, the security of the scheme holds in the $\mathbf{HUF}^{\mathrm{e,s}}_{\mathrm{MB}}$ model.

On the ◆ marks of Fig. 6. The ◆ marks in Fig. 6 refer to instantiations where neither Theorem 1 nor Theorem 2 apply. In other words, it is neither possible to efficiently extract the full ZK-PoK commitment R, nor possible to have the alignment property. For instance, this is reflected by having part of R hashed into cv of hs (namely the x coordinate R_{x} and a prefix of the y coordinate R_y), and a very few bits of the y coordinate R_y stored in the buffer in clear. The instances marked with ◆ require further cryptanalysis in order to (possibly) be deemed broken.

A More on Vertical Session Management Arrows

As an illustration of the semantics of the vertical session management arrows, we provide an equivalent explicit version of Figs. 12 in 17. Note that many extra lines of code are required to (1) implement a state machine for each session that ensures the oracles are invoked strictly in the right order, and (2) implement the persistence of state variables of each session. These changes implement precisely what the (now dashed) vertical arrows promise to be doing.

B Details Deferred from Sect. 5

We provide further information on our attack against the Schnorr ZK-PoK protocol. All line references are with respect to Fig. 13.

Description of Subroutines. $\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3$. Regarding A_1 observe that the λ_i defined in line 16 are (Lagrange) polynomials $\mathbb{Z}_p^l \to \mathbb{Z}_p$ that have, for all $1 \leq i \leq l$ and $b \in \{0,1\}$, the property $x_i = c_i^b \implies \lambda_i(\ldots x_i \ldots) = b$. Polynomial P is defined in line 17 as a weighted sum of the λ_i's; as all the latter are degree-one polynomials, finding suitable coefficients $\alpha_0, \ldots, \alpha_l$ in line 18 is a well-defined operation and always possible. See the caption of Fig. 13 for a strategy for how the weights w_i can be chosen such that line 22 always succeeds efficiently. The remaining parts of $\mathrm{A}_1, \mathrm{A}_2, \mathrm{A}_3$ should be clear from the instructions.

Proof of Validity of Transcript τ^* of Line 13. We need to argue that $g^{s^*} = R^* X^{c^*}$. Observe that, once line 13 of adversary A is reached, for the values $\alpha_i, \bar{c}_i, b_i, c^*$ we have

$$\alpha_0 + \textstyle\sum_{i=1}^l \alpha_i \bar{c}_i = P(\bar{c}_1 \mathinner{..} \bar{c}_l) = \sum_{i=1}^l w_i \lambda_i(\bar{c}_1 \mathinner{..} \bar{c}_l) = \sum_{i=1}^l w_i b_i = \alpha_0 + c^* \quad , \quad (1)$$

where the first, second, and fourth equalities are by lines 18, 17, and 22, respectively. (The third equality was already discussed.) This implies

$$g^{s^*} = \prod (g^{s_i})^{\alpha_i} = \prod (R_i X^{\bar{c}_i})^{\alpha_i} = R^* X^{\Sigma \alpha_i \bar{c}_i} = R^* X^{c^*} \quad ,$$

```
INITIALIZATIONS: $A, A^{\cdot}, B, B^{\cdot}, V, V^{\cdot}, W, W^{\cdot} \leftarrow \emptyset$; $SST[\cdot] \leftarrow 0$; $ST[\cdot], HS[\cdot], HV[\cdot], M[\cdot], S[\cdot] \leftarrow \diamond$

Game $\mathbf{UF}^{e,s}_{BB}(\mathcal{A})$
00 $sk, vk \leftarrow$ gen
01 $\mathcal{A}(vk)$
02$^{e}$ Promise $V \subseteq A$
03$^{s}$ Promise $V^{\cdot} \subseteq A^{\cdot}$
04 Lose

Game $\mathbf{HUF}^{e,s}_{BB}(\mathcal{A})$
05 $sk, vk \leftarrow$ gen
06 $\mathcal{A}(vk)$
07$^{e}$ Promise $W \subseteq B$
08$^{s}$ Promise $W^{\cdot} \subseteq B^{\cdot}$
09 Lose

Oracle SgnIni($sid$)
10 Require $SST[sid] = 0$
11 $SST[sid] \leftarrow \diamond$
12 $st, hs \leftarrow$ sgn.ini($sk$)
13 Share $hs$
   : : : : : : :
14 $ST[sid] \leftarrow st$
15 $HS[sid] \leftarrow hs$
16 $SST[sid] \leftarrow (s, 1)$

Oracle SgnHash($sid, m$)
17 Require $SST[sid] = (s, 1)$
18 $SST[sid] \leftarrow \diamond$
19 $hs \leftarrow HS[sid]$
20 $hv \leftarrow hash(hs, m)$
21 Share $hv$
22 $HV[sid] \leftarrow hv$
23 $M[sid] \leftarrow m$
24 $SST[sid] \leftarrow (s, 2)$

Oracle SgnFin($sid, \overline{hv}$)
25 Require $SST[sid] = (s, 2)$
26 $SST[sid] \leftarrow \diamond$
27 $st \leftarrow ST[sid]$
28 $hs \leftarrow HS[sid]$
29 $hv \leftarrow HV[sid]$
30 $m \leftarrow M[sid]$
31 Require $\overline{hv} = hv$
32 $\sigma \leftarrow$ sgn.fin($st, hv$)
33 Share $\sigma$
34 $A \overset{\cup}{\leftarrow} \{m\}$
35 $A^{\cdot} \overset{\cup}{\leftarrow} \{(m, \sigma)\}$
36 $B \overset{\cup}{\leftarrow} \{(hs, hv)\}$
37 $B^{\cdot} \overset{\cup}{\leftarrow} \{(hs, hv, \sigma)\}$

Oracle VfyIni($sid, \sigma$)
38 Require $SST[sid] = 0$
39 $SST[sid] \leftarrow \diamond$
40 $st, hs \leftarrow$ vfy.ini($vk, \sigma$)
41 Share $hs$
42 $S[sid] \leftarrow \sigma$
43 $ST[sid] \leftarrow st$
44 $HS[sid] \leftarrow hs$
45 $SST[sid] \leftarrow (v, 1)$

Oracle VfyHash($sid, m$)
46 Require $SST[sid] = (v, 1)$
47 $SST[sid] \leftarrow \diamond$
48 $hs \leftarrow HS[sid]$
49 $hv \leftarrow hash(hs, m)$
50 Share $hv$
51 $HV[sid] \leftarrow hv$
52 $M[sid] \leftarrow m$
53 $SST[sid] \leftarrow (v, 2)$

Oracle VfyFin($sid, \overline{hv}$)
54 Require $SST[sid] = (v, 2)$
55 $SST[sid] \leftarrow \diamond$
56 $st \leftarrow ST[sid]$
57 $hs \leftarrow HS[sid]$
58 $hv \leftarrow HV[sid]$
59 $\sigma \leftarrow S[sid]$; $m \leftarrow M[sid]$
60 Require $\overline{hv} = hv$
61 $v \leftarrow$ vfy.fin($st, hv$)
62 Share $v$
63 If $v$: $V \overset{\cup}{\leftarrow} \{m\}$
64 If $v$: $V^{\cdot} \overset{\cup}{\leftarrow} \{(m, \sigma)\}$
65 If $v$: $W \overset{\cup}{\leftarrow} \{(hs, hv)\}$
66 If $v$: $W^{\cdot} \overset{\cup}{\leftarrow} \{(hs, hv, \sigma)\}$
```

Fig. 17. Explicit version of Fig. 12. The colons between lines 13,14 serve no purpose beyond visually aligning the middle and right columns. Array SST stores session states.

where the first and third equalities are by lines 24 and 19, respectively, the second equality follows from the validity of the τ_i, and the fourth equality follows from (1). Overall this shows the claim. $\square$

ROS PROBLEM. Purely for reference we recall the definition of the ROS problem. Note that $\langle \cdot , \cdot \rangle$ stands for the (inner) dot product.

Definition 4 (ROS Problem [11]). *Given a prime number p, a positive integer l, and a random oracle $H : \mathbb{Z}_p^l \to \mathbb{Z}_p$, find vectors $v_0, \ldots, v_l, c \in \mathbb{Z}_p^l$ such that $i \neq j \Rightarrow v_i \neq v_j$ and $0 \leq i \leq l \Rightarrow \langle v_i, c \rangle = H(v_i)$.*

References

1. PyCryptodome's documentation. https://pycryptodome.readthedocs.io/en/latest/src/signature/signature.html
2. Secure hashes and message digests. https://docs.python.org/3/library/hashlib.html
3. PKCS#11: Cryptographic Token Interface Standard. An RSA Laboratories Technical Note (1995), https://github.com/Pkcs11Interop/PKCS11-SPECS/blob/master/v1.0/pkcs-11.pdf
4. Elliptic curve cryptography (June 2018), https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TR03111/BSI-TR-03111_V-2-1_pdf.pdf?__blob=publicationFile&v=2
5. IT Security techniques - Digital signatures with appendix - Part 3: Discrete logarithm based mechanisms. ISO14888-3 (2018)
6. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU (2020), https://falcon-sign.info/falcon.pdf
7. Guide de sélection d'algorithmes cryptographiques. Guide ANSSI (March 2021), https://www.ssi.gouv.fr/uploads/2021/03/anssi-guide-selection_crypto-1.0.pdf
8. Digital signature standard (DSS). FIPS 186-5 (February 2023), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf
9. Module-Lattice-Based Digital Signature Standard. Federal Information Processing Standards Publication (2023), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.ipd.pdf
10. Backes, M., Barthe, G., Berg, M., Grégoire, B., Kunz, C., Skoruppa, M., Zanella-Béguelin, S.: Verified security of merkle-Damgård. In: Zdancewic, S., Cortier, V. (eds.) CSF 2012 Computer Security Foundations Symposium. pp. 354–368. IEEE Computer Society Press (2012). https://doi.org/10.1109/CSF.2012.14
11. 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
12. Coron, J.S., Dodis, Y., Malinaud, C., Puniya, P.: Merkle-Damgård revisited: How to construct a hash function. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 430–448. Springer, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_26
13. Finney, H., Donnerhacke, L., Callas, J., Thayer, R.L., Shaw, D.: OpenPGP Message Format. RFC 4880 (Nov 2007). https://doi.org/10.17487/RFC4880, https://www.rfc-editor.org/info/rfc4880
14. GnuPG: Cryptographic Functions. https://www.gnupg.org/documentation/manuals/gcrypt/Cryptographic-Functions.html#index-gcry_005fpk_005fhash_005fsign
15. Google: BoringSSL. https://commondatastorage.googleapis.com/chromium-boringssl-docs/ecdsa.h.html#Signing-and-verifying
16. Housley, R.: Cryptographic Message Syntax (CMS). RFC 5652 (Sep 2009). https://doi.org/10.17487/RFC5652, https://www.rfc-editor.org/info/rfc5652
17. Jonsson, J., Kaliski, B.: Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447 (Feb 2003). https://doi.org/10.17487/RFC3447
18. Josefsson, S., Liusvaara, I.: Edwards-Curve Digital Signature Algorithm (EdDSA). RFC 8032 (Jan 2017). https://doi.org/10.17487/RFC8032, https://www.rfc-editor.org/info/rfc8032

19. Ounsworth, M.: Design rationale for keyed message digests in SPHINCS+, Dilithium, FALCON? https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/cIsc6tUY9Rw
20. Ounsworth, M.: Whether to hash-then-sign with Dilithium and Falcon? https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/yg9z4keaEf4
21. Poettering, B., Rastikian, S.: Formalizing hash-then-sign signatures. In: Public Key Cryptography (1). LNCS, vol. 14601, pp. 289–315. Springer (2024), https://link.springer.com/chapter/10.1007/978-3-031-57718-5_10
22. Schneier, B.: Applied Cryptography. John Wiley & Sons, New York, second edn. (1996)
23. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0_22
24. Tadahiko Ito (SECOM CO., L.: Considerations on separation of hash. https://datatracker.ietf.org/meeting/interim-2021-lamps-01/materials/slides-interim-2021-lamps-01-sessa-position-presentation-by-tadahiko-ito-00
25. Wuille, P., Nick, J., Ruffing, T.: Schnorr Signatures for secp256k1. BIP 340 (Jan 2020), https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki
26. Zimman, C., Bong, D.: Pkcs #11 cryptographic token interface current mechanisms specification version 3.0 (June 2020), https://docs.oasis-open.org/pkcs11/pkcs11-curr/v3.0/os/pkcs11-curr-v3.0-os.html

Adaptor Signatures: New Security Definition and a Generic Construction for NP Relations

Xiangyu Liu[1,2](✉), Ioannis Tzannetos[1,3], and Vassilis Zikas[2]

[1] Purdue University, West Lafayette, USA
{liu3894,itzannet}@purdue.edu
[2] Georgia Institute of Technology, Atlanta, USA
vzikas@gatech.edu
[3] National Technical University of Athens, Athens, Greece

Abstract. An adaptor signatures (AS) scheme is an extension of digital signatures that allows the signer to generate a *pre-signature* for an instance of a hard relation. This pre-signature can later be adapted to a full signature with a corresponding witness. Meanwhile, the signer can extract a witness from both the pre-signature and the signature. AS have recently garnered more attention due to its scalability and interoperability. Dai *et al.* [INDOCRYPT 2022] proved that AS can be constructed for any NP relation using a generic construction. However, their construction has a shortcoming: the associated witness is exposed by the adapted signature. This flaw poses limits the applications of AS, even in its motivating setting, i.e., blockchain, where the adapted signature is typically uploaded to the blockchain and is public to everyone.

To address this issue, in this work we augment the security definition of AS by a natural property which we call *witness hiding*. We then prove the existence of AS for any NP relation, assuming the existence of one-way functions. Concretely, we propose a generic construction of witness-hiding AS from signatures and a weak variant of trapdoor commitments, which we term *trapdoor commitments with a specific adaptable message*. We instantiate the latter based on the Hamiltonian cycle problem. Since the Hamiltonian cycle problem is NP-complete, we can obtain witness hiding adaptor signatures for any NP relation.

1 Introduction

Blockchain technology has emerged as a disruptor, offering decentralized frameworks for various applications. Each transaction on the blockchain operates within a scripting language validated by nodes through a decentralized consensus protocol. Cryptocurrencies like Bitcoin and Ethereum utilize blockchain technologies to power their operations. However, executing transactions on blockchains often incurs significant costs, as users are required to pay fees to

Work done while the authors were at Purdue University.

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15485, pp. 168–193, 2025.
https://doi.org/10.1007/978-981-96-0888-1_6

entities that run the consensus protocol. These fees are determined by the storage and computational costs associated with transaction scripts. To mitigate this issue, the utilization of adaptor signatures has been proposed as a means to reduce on-chain fees paid to nodes in a wide range of decentralized finance (DeFi) applications (see some examples below).

The notation of adaptor signatures (AS, a.k.a. scriptless scripts) was proposed by Poelstra in 2017 [22,23] and later formalized by Aumayr *et al.* [2,3]. An AS scheme is related to a hard relation R such that the signer, holding the signing secret key, can pre-sign a message (e.g., a transaction) with respect to some instance Y to obtain a pre-signature $\tilde{\sigma}$, which can later be adapted to a full signature σ with the knowledge of y, the witness of Y such that $(Y, y) \in R$. Moreover, from both the pre-signature $\tilde{\sigma}$ and the adapted signature σ, the signer can extract a witness of Y. AS can be viewed as an extension of (ordinary) signatures by additionally addressing mutual trust between the signer and the receiver, since the secret witness is exposed to the signer once the full signature has been published.

AS are widely applied in fair exchanges [8], atomic swaps [13,23], and payment channel networks [2,13] to reduce on-chain computations and improve the fungibility of transactions. We briefly discuss the applications of AS as follows.

Fair Exchange of a Witness. Assume Alice, who holds a token c for some cryptocurrency, e.g., some amount of ETH for Ethereum, wants to trade it for a witness y of some instance Y held by Bob (where y may be some secret information accessing some digital services). Alice can post to the blockchain a timeout transaction transferring c to Bob, which however requires a full AS signature (with respect to Y) to be claimed; then, off-chain, Alice can pre-sign a transaction tx using Y and send the pre-signature $\tilde{\sigma}$ to Bob. AS allow then Bob to adapt $\tilde{\sigma}$ to a full signature σ using y, and upload it to the blockchain to receive c. Once σ has been published, Alice can extract the witness y which completes the exchange.

Atomic Swaps. Atomic swaps [13,23] allow two parties, Alice and Bob, to exchange assets in two different cryptocurrencies c_A and c_B. First, both Alice and Bob lock c_A and c_B on the blockchain as deposits (a.k.a. collateral). Then, Alice randomly samples an instance-witness pair (Y, y), generates a pre-signature $\tilde{\sigma}_A$ on message tx_A and instance Y, and sends tx_A, Y, $\tilde{\sigma}_A$ to Bob. Here, tx_A is a transaction for transferring c_A to Bob. Then, Bob also generates a pre-signature $\tilde{\sigma}_B$ on message tx_B and instance Y, and sends tx_B, $\tilde{\sigma}_B$ to Alice, where tx_B is a transaction for transferring c_B to Alice. After receiving $\tilde{\sigma}_B$, Alice can adapt it to a full signature σ_B with the knowledge of y, and upload it to the blockchain to obtain c_B. Meanwhile, Bob can also extract y from $\tilde{\sigma}_B$ and σ_B, adapt $\tilde{\sigma}_A$ to σ_A, and hence obtain c_A.

Multi-hop Payments. Multi-hop payments [13] allow multiple parties to route payments between them, provided that they have a payment channel with a common intermediate[1]. Consider four parties Alice, Bob, Charlie

[1] In the original protocol in [13], each party is sampling a new pair of instance-witness and they are using it once for each payment. Here we simplify the protocol by

and David, where Alice wants to pay cryptocurrency (say c) to David. First, Alice and Bob lock some funds on the blockchain on a payment channel as deposits/collateral, Bob with Charlie, and Charlie with David do likewise. Then, David randomly samples an instance/witness pair (Y, y) and forwards Y to Alice, Bob, and Charlie. Subsequently, Alice generates a pre-signature $\tilde{\sigma}_A$ on message tx_A and instance Y, and then sends $tx_A, \tilde{\sigma}_A$ to Bob. Here tx_A is a transaction for transferring c to Bob. Then Bob also generates a pre-signature $\tilde{\sigma}_B$ on message tx_B (transaction for transferring c to Charlie) and instance Y and sends $tx_B, \tilde{\sigma}_B$ to Charlie. After that, Charlie generates a pre-signature $\tilde{\sigma}_C$ on message tx_C (transaction for transferring c to David) and instance Y, and sends $tx_C, \tilde{\sigma}_C$ to David. After receiving $\tilde{\sigma}_C$, David can adapt it to a full signature σ_C with the knowledge of y, and upload it to the blockchain to obtain c. Meanwhile, Charlie can also extract y from $\tilde{\sigma}_C$ and σ_C, adapt $\tilde{\sigma}_B$ to σ_B and hence obtain c. Finally Bob can follow the same procedure to obtain c.

Security of Adaptor Signatures. The security definition of AS was formalized by Aumayr *et al.* [2,3] and adopted by almost all subsequent works[2] ([12,13,18,26–28], to name a few). We give the formal security definition of AS in Sect. 2.3. Nonetheless, as finding an issue with the existing definition (and repairing it) is one of our contributions, we provide here an informal discussion.

As an extension of signatures, AS inherits the classical unforgeability property of signatures. Namely, only the owner of the secret key can generate a valid pre-signature (and a regular signature, of course). Besides classical unforgeability, two additional properties are required for the security of the sender and the receiver.

Security for the the Sender (a.k.a. Witness Extractability). The sender can extract a witness from the valid pre-signature and the valid adapted signature.

Security for the Receiver (a.k.a. Pre-signature Adaptability). The receiver can adapt a valid pre-signature into a valid (full) signature with the knowledge of a witness.

Notice that an AS scheme is defined with respect to a hard relation R, which can vary from simple discrete logarithm relations to more complex relations based on a blockchain scripting language. Therefore, a natural question is:

What relation R can an adaptor signature scheme support?

allowing every party to take the same instance. The security still holds assuming that the intermediate parties do not collude.

[2] Dai *et al.* [10] identified that Aumayr *et al.*'s definition [2,3] does not consider the case of multiple pre-sign queries by the adversary, and fixed it by proposing a so-called full extractability property.

Adaptor Signatures for NP Relations. Most previous works [2,13,18,26,28] focus on constructing AS schemes based on particular signatures schemes (like the ECDSA and Schnorr) and for script-related relations (like the public/secret key relation of signatures). The more recent work by Dai, Okamoto, and Yamamoto [10] gave a generic constructions of AS for general NP relations. They showed that AS can be constructed from any signature scheme and any NP-hard relation, and therefore, adaptor signatures are implied by one-way functions.

An advantage of the construction from [10] is its simplicity: Let SIG be a signature scheme, and R be an NP relation. In Dai *et al.*'s generic construction GAS1, a pre-signature of message m w.r.t. instance Y is in the form of $\tilde{\sigma} = (\bar{\sigma}, Y)$, where $\bar{\sigma}$ is a signature of SIG for message (m, Y). To adapt $\tilde{\sigma}$ into a full signature, one just attaches the witness y to $\tilde{\sigma}$ and obtains $\sigma = (\bar{\sigma}, Y, y)$. The verification algorithm of AS checks both the validity of $\bar{\sigma}$ w.r.t. message (m, Y) and that $(Y, y) \in R$.

However, as it turns out the above simplicity comes at a high cost: the witness y is exposed in the adapted signature in plain text. This poses serious security risks in many applications. For example, in the fair exchange application above, the adapted signature σ is uploaded to the blockchain, making the witness accessible to everyone on the network. However, y should only be known to the buyer (Alice) since she has made a payment to the seller (Bob).

Similar security issues also arise in multi-hop payments when the construction by [10] is used. Consider the case that Alice wants to pay to David via two intermediary nodes, Bob and Charlie. If y is contained in plain in an adapted signature, then after David uploads σ_C, Bob is able to get y and adapt $\tilde{\sigma}_A$ into σ_A, thus skipping Charlie and receiving his money, which is conflict with the fairness of multi-hop payments.

As it turns out, the above issue is not just an issue of the construction in [10], but rather a deeper issue with the definition of security of AS. Intuitively, the functionality of AS requires that a witness can be extracted from *both* the pre-signature $\tilde{\sigma}$ and the adapted signature σ, but not from either of them individually. In almost all existing AS schemes [2,11,13,18,28], both $\tilde{\sigma}$ and σ are essential for extracting a witness. However, the generic construction GAS1 from [10] satisfies all security requirements of the formal AS security definition by Aumayr *et al.* [2,3]—including unforgeability, witness extractability, and pre-signature adaptability—but still fails in satisfying the above intuition. This demonstrates that previous security definition does not cover all security properties needed for the applications of AS, and points to a new hole in the literature (which we fill in) of a generic AS construction for any NP relation.

To solve the first (definitional) problem, we introduce a new security property called *witness hiding*. Informally, it requires that the witness y can be extracted from both a pre-signature and an adapted signature (jointly), but not from only one of them alone. Thus the key open question now is:

Question: *Can witness hiding adaptor signatures support any NP relation, and, if so, what is the minimal assumption for such a construction?*

In this work, we propose a generic construction of AS from any signature scheme and for any NP relation. Since signatures can be constructed from one-way functions [15], we obtain the following theorem.

Theorem 1. *Assuming one-way functions exist, then there exist witness hiding adaptor signatures for any NP language.*

We summarize our contributions as follows:

– We introduce witness hiding, a new security property for adaptor signatures. This property requires that the witness y can be extracted from both a pre-signature and an adapted signature, but not from only one of them. Witness hiding is crucial in most applications of AS, including fair exchanges [8], atomic swaps [13,23], and payment channel networks [2,13], where the pre-signature remains private while the adapted signature is uploaded to the blockchain and is public to everyone. Witness hiding helps prevent an eavesdropper from extracting a witness from only the adapted signature. We observe that the only existing adaptor signature scheme for any NP relation [10] does not satisfy witness hiding, as the witness is exposed in plain in the adapted signature[3].
– We propose a generic construction of witness-hiding adaptor signatures from (ordinary) signatures and a new type of trapdor commitment which we term *trapdoor commitments with a specific adaptable message.* The latter is a weaker version of classical trapdoor commitments, where there is a specific message $m_{\mathbf{0}}$, and with the knowledge of the trapdoor, one can open a commitment of $m_{\mathbf{0}}$ to another message m.
– We propose a trapdoor commitment scheme with a specific adaptable message based on the Hamiltonian cycle problem, where the commitment key is the Hamiltonian problem instance and the trapdoor is a Hamiltonian cycle (witness). Since the Hamiltonian cycle problem is NP-complete, we obtain adaptor signatures for any NP language. See Fig. 1 for a framework of adaptor signatures.

1.1 Related Work

The concept of adaptor signatures (AS) was introduced by Poelstra [22,23] (referred as scriptless scripts in [22]). In 2020, Aumayr *et al.* [2,3] first formalized adaptor signatures, and proposed three security properties for AS: unforgeability, pre-signature adaptability, and witness extractability. All subsequent follow-up works on AS can be categorized into two main directions.

The first direction focuses on designing AS from known underlying signature schemes. For example, the ECDSA-based adaptor signature scheme [2], the Schnorr-based scheme [2,28], the LWE/SIS-based scheme LAS [13], the isogeny-based scheme IAS [26], the code-based scheme [18], etc. Note that the supported

[3] Almost all previous constructions ([2,11,13,18,26,28]), except GAS1 [10] satisfy witness hiding property.

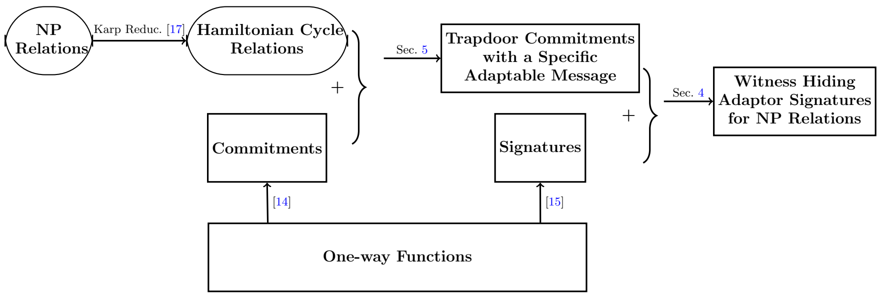

NP Relations
Karp Reduc. [17]
Hamiltonian Cycle Relations
+
Commitments
Sec. 5
Trapdoor Commitments with a Specific Adaptable Message
+
Signatures
Sec. 4
Witness Hiding Adaptor Signatures for NP Relations
[14]
[15]
One-way Functions

Fig. 1. A framework of witness hiding adaptor signatures for NP relations

languages (e.g., the discrete logarithm language) in these schemes are fixed, due to the specific structures of the underlying signature schemes.

The second direction focuses on generic constructions of AS [10,11]. Erwig *et al.* [11] showed that identification (ID) schemes with additional homomorphic properties can be transformed into adaptor signatures. However, the transform requires the supporting language to be highly related with the format of the commitment in the ID scheme, limiting the instantiations to the DL-based or the RSA-based ID schemes and their corresponding languages.

Dai *et al.* [10] proposed the first truly generic construction of AS, called GAS1, from any signature scheme and any NP language. As discussed however the (overly-)simple construction GAS1, despite satisfying the security requirements according to [2], has the significant issue: that the witness y is exposed in the adapted signature, which poses a serious security vulnerability in blockchain applications.

In fact, [10] also proposed a second generic construction called GAS2 from any signature scheme and any strongly random-self reducible relation. Compared to GAS1, GAS2 is unlinkable, i.e., the adapted signature is indistinguishable from a normally generated signature, and hence the witness is hidden from only the adapted one. However, GAS2 requires the strong random-self reducibility of the underlying relation. Therefore, similar to [11], the instantiations of GAS2 are limited to standard number theoretical problems such as DL, RSA, and LWE.

1.2 Technical Overview

In this subsection, we provide a brief overview of our techniques.

Defining Witness Hiding. The witness hiding property requires that the witness y is extractable from both the pre-signature $\tilde{\sigma}$ and the adapted signature σ (jointly), but not from either of them alone. Note that y is inherently hidden in $\tilde{\sigma}$ since the pre-sign algorithms takes only the message, the instance and the secret key as inputs, and it is independent of y. More formally, we say an AS scheme has the witness hiding property, if there exists an simulator that, given the secret key (of AS), the message, and the instance as inputs, outputs a signature which is indistinguishable from a signature adapted from a pre-signature using witness y.

Generic Construction. Next we illustrate our generic construction of AS from any signature scheme and for any NP relation. Our approach draws inspiration from the simplicity of [10] but significantly modifies their paradigm with novel ideas to achieve the witness-hiding property.

More concretely, let SIG be an ordinary signature scheme and R be a hard relation. Recall that in GAS1, a pre-signature for message m and instance Y is in the form of $\tilde{\sigma} = (\bar{\sigma}, Y)$, where $\bar{\sigma}$ is a signature of SIG for message (m, Y). And the adapted signature is just $\sigma = (\bar{\sigma}, Y, y)$ by attaching the witness to $\tilde{\sigma}$.

One might be tempted to use the following idea to avoid the exposure of y in σ: replace y with a zero-knowledge proof, and show that the adaptor knows a witness of Y. However, in such a modification, y cannot be extracted from both $\tilde{\sigma}$ and σ, and the witness extractability is violated.

Our key insight here is the observation that the witness extractability property of AS has similarities in spirit with the special soundness property of Sigma protocols. To demonstrate this, let us first recall this property. A Sigma protocol for a hard relation R is a three-move protocol between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, where the prover holds a witness y of some instance Y and wants to prove to the verifier in a zero-knowledge way. A complete transcript of a protocol execution consists of three parts: the first move is a commitment cmt sent from $\mathcal{P}$ to $\mathcal{V}$; the second move is a random challenge ch from $\mathcal{V}$ to $\mathcal{P}$; and the third move is a response rsp from $\mathcal{P}$ to $\mathcal{V}$.

- **Special soundness of Sigma protocols.** From two valid transcripts with the same commitment but different challenges, one can extract a witness of the instance.
- **Witness extractability of AS.** From a valid pre-signature and an adapted signature one can extract a witness.

Inspired by this analogy observation, we modify the pre-signature of AS to be $\tilde{\sigma} = (\bar{\sigma}, Y, (cmt, ch', rsp'))$, where $\bar{\sigma}$ is a signature of (m, Y, cmt), and (cmt, ch', rsp') is a valid transcript of a Sigma protocol w.r.t. instance Y. To adapt it to a (full) signature, the adaptor, with the knowledge of y, has to generate rsp for another challenge $ch \neq ch'$ such that (cmt, ch, rsp) is also a valid transcript. This is feasible due to the completeness of the Sigma protocol, since a prover knowing a witness y is able to answer any challenge and reply a response to make the transcript valid. Meanwhile, the witness extractability of AS is guaranteed by the special soundness of the Sigma protocol, since from $\tilde{\sigma} = (\bar{\sigma}, Y, (cmt, ch, rsp))$ and $\sigma = (\bar{\sigma}, Y, (cmt, ch' \neq ch, rsp'))$ one can extract a witness of Y.

We formally describe our generic construction using trapdoor commitments (TC, a.k.a. chameleon hashes [19], which are equivalent to Sigma protocols [4]).

In a TC scheme, with the public commitment key, one can commit a message m' to get a commitment c and an opening d', and with the trapdoor one can open c to another $m \neq m'$ and get a corresponding d. Meanwhile, trapdoor extractability requires that from a collision (c, m', d') and $(c, m \neq m', d)$ one can extract the trapdoor. In our generic construction of AS, we first transfer the instance-witness pair (Y, y) into a commitment-trapdoor key pair of a TC scheme. Now, a pre-signature w.r.t. message m and instance Y is in the form of

$$\tilde{\sigma} = (\bar{\sigma}, Y, c, m' \neq m, d'),$$

where $\bar{\sigma}$ is a signature for (m, Y, c), and d' is an opening of c for a "dummy" message $m' \neq m$. Given $\tilde{\sigma}$ and witness y (the trapdoor of the underlying TC scheme), the adapted signature is in the form of

$$\sigma = (\bar{\sigma}, Y, c, m, d),$$

where d is another opening for the signed message m, adapted from (c, m', d') using trapdoor (witness) y.

Functionality and security of the generic construction are analyzed as follows.

- **Functionality of adaption.** This is guaranteed by the trapdoor adaption property of TC.
- **Unforgeability.** This is inherited from the underlying signature scheme.
- **Witness extractability.** This is guaranteed by the trapdoor extractability of TC. Namely, from a collision (c, m', d') and $(c, m \neq m', d)$ one can extract the trapdoor.
- **Pre-signature adaptability.** This is guaranteed by the functionality of TC, i.e., with the trapdoor one can open a commitment to any message, and therefore the adapted signature $\sigma = (\bar{\sigma}, Y, c, m, d)$ is valid.
- **Witness hiding.** This is due to the fact that from only a tuple of commitment (c, m, d) nothing about the trapdoor is leaked.

We notice that in the pre-sign process, the dummy message m' in the pre-signature can be a fixed value $m_{\mathbf{0}}$, as long as it differs from the message m to be signed. Moreover, to construct AS schemes, we only require a property that a commitment of the fixed $m_{\mathbf{0}}$ (but not necessarily an arbitrary commitment) can be opened to another message. Based on this observation, we propose a weakened notation of trapdoor commitments, termed trapdoor commitments with a specific adaptable message, where there exists a specific message $m_{\mathbf{0}}$, and with the trapdoor one can (only) open a commitment of $m_{\mathbf{0}}$ to another message. Next we will see, such a weakening enables constructions from any NP relation, where the instance Y and the witness y serve as the commitment key and the trapdoor, respectively.

Constructing TC with a Specific Adaptable Message. We now turn to construct a trapdoor commitment scheme with a specific adaptable message for any NP relation R. Bellare and Ristov [4] proved the equivalence of Sigma protocols and chameleon hashes (and hence trapdoor commitments), where the commitment, the challenge, and the response in a Sigma protocol correspond to the commitment, the message, and the opening in a trapdoor commitment scheme. However, one must exercise caution when transferring one to another, since their security definitions are not perfectly matched. For example, to transfer a Sigma protocol into a trapdoor commitment scheme, we must additionally ensure that there is a simulator for the Sigma protocol, which can generate a simulated transcript given a fixed challenge, and the commitment can be recovered from the challenge, the response, and the instance. However, this is not a universally applicable property for all Sigma protocols.

Following the framework by Bellare and Ristov [4], we found that the Sigma (zero-knowledge) protocol for the Hamiltonian cycle problem by Blum [7,14] can be perfectly transferred into a trapdoor commitment scheme with a specific adaptable message $m_\mathbf{0} = 0$. We recall the protocol and present the corresponding trapdoor commitment scheme in Fig. 2.

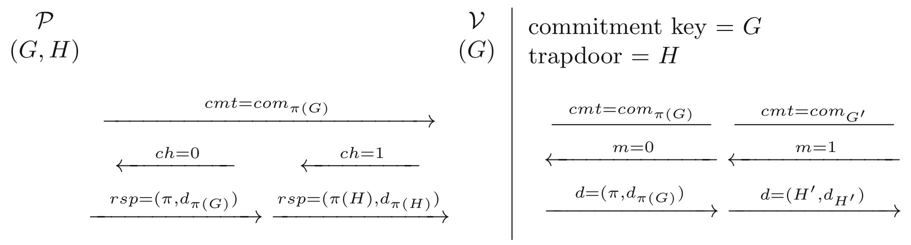


Fig. 2. The zero-knowledge proof protocol for the Hamiltonian cycle problem [7,14] (left) and the trapdoor commitment scheme from it (right). Here $(G, H \subseteq G)$ is an instance-witness pair of the Hamiltonian cycle problem, and *com* and d are commitments and openings of a bit commitment scheme with statistical biding and computational hiding.

Let G be a graph, and $H \subseteq G$ be a Hamiltonian cycle, i.e., a witness of Hamiltonian graph instance G. First, the prover $\mathcal{P}$ randomly samples a permutation π and commits $G' = \pi(G)$, and then sends the commitments $com_{G'}$[4] to the verifier $\mathcal{V}$. Here *com* denotes standard commitments with statistical biding and computational hiding (cf. Definition 1). Then the verifier $\mathcal{V}$ sends a random challenge $ch \xleftarrow{\$} \{0, 1\}$. If $ch = 0$, then $\mathcal{P}$ sends all openings of $com_{G'}$ and the permutation π to $\mathcal{V}$, and $\mathcal{V}$ checks $com_{G'}$ are commitments of $\pi(G)$. And if $ch = 1$, then $\mathcal{P}$ sends all openings of $com_{H'}$ to $\mathcal{V}$, and $\mathcal{V}$ checks $com_{G'}$ include commitments of a Hamiltonian cycle $H'(= \pi(H))$.

The Sigma protocol described above has a zero-knowledge simulator that, given the challenge ch, can perfectly simulate a transcript (cmt, ch, rsp). Moreover, if $ch = 0$, then the simulated transcript is identical to the transcript from

[4] More precisely, $com_{G'}$ is a group of bit commitments for the adjacency matrix of G'.

an honest execution. And with the knowledge of a witness, it is easy to get a response for $ch = 1$ under the same commitment, which is exactly the functionality of adaption in the trapdoor commitment scheme.

Our trapdoor commitment scheme with specific adaptable message $m_{\mathbf{0}} = 0$ is shown in Fig. 2 (right). If $m = 0$, then the commitment is $com_{\pi(G)}$ and the corresponding opening is $(\pi, d_{\pi(G)})$, where π is a random permutation and $d_{\pi(G)}$ is the corresponding openings of the underlying (standard) commitment scheme. If $m = 1$, then the commitment is $com_{G'}$ and the corresponding opening is $d_{H'}$, where G' is a randomly generated Hamiltonian graph with a Hamiltonian cycle H'.

Given that the Hamiltonian cycle problem is NP-complete [17], we know any NP relation R can be transferred into a trapdoor commitment scheme with a specific adaptable message. Therefore, we get witness hiding adaptor signature schemes from any signature scheme and for any NP relation. Combined with the fact that signature schemes and (standard) bit commitment schemes are implied by one-way functions [14,15], Theorem 1 holds consequently.

1.3 Organization of the Paper

This rest of the paper is organized as follows. In Sect. 2, we present preliminaries and define trapdoor commitments with a specific adaptable message. In Sect. 3, we introduce adaptor signatures and their security properties, including the witness hiding property. Section 4 details the generic construction of AS. The trapdoor commitment scheme with a specific adaptable message for the Hamiltonian cycle problem is shown in Sect. 5. Finally, we conclude this paper in Sect. 6.

2 Preliminaries

Throughout this paper, we use $\lambda \in \mathbb{N}$ to denote the security parameter. For $\mu \in \mathbb{N}$, define $[\mu] := \{1, 2, ..., \mu\}$. Denote by $x := y$ the operation of assigning y to x. Denote by $x \overset{\$}{\leftarrow} \mathcal{S}$ the operation of sampling x uniformly at random from a set $\mathcal{S}$. For a distribution $\mathcal{D}$, denote by $x \leftarrow \mathcal{D}$ the operation of sampling x according to $\mathcal{D}$. For an algorithm $\mathcal{A}$, denote by $y \leftarrow \mathcal{A}(x; r)$, or simply $y \leftarrow \mathcal{A}(x)$, the operation of running $\mathcal{A}$ with input x and randomness r and assigning the output to y. For deterministic algorithms $\mathcal{A}$, we also write as $y := \mathcal{A}(x)$ or $y := \mathcal{A}(x; r)$. "PPT" is short for probabilistic polynomial-time.

2.1 Commitments

Definition 1 (Commitments). *A commitment scheme consists of the following three algorithms. Namely,* $\mathsf{COM} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Ver})$.

- $ck \leftarrow \mathsf{Gen}(1^\lambda)$. *The key generation algorithm takes as input the security parameter* λ, *and outputs a commitment key* ck.

- $(c, d) \leftarrow \mathsf{Com}(ck, m)$. *The commitment algorithm takes as input ck and a message $m \in \mathcal{M}$, and outputs a commitment c and an opening d.*
- $0/1 \leftarrow \mathsf{Ver}(ck, c, m, d)$. *The verification algorithm takes as input ck, c, m and d, and outputs a bit.*

Correctness. *For any $ck \leftarrow \mathsf{Gen}(1^\lambda)$, any message $m \in \mathcal{M}$ and $(c, d) \leftarrow \mathsf{Com}(ck, m)$, it holds that $\mathsf{Ver}(ck, c, m, d) = 1$.*

Definition 2 (Statistical Biding of Commitments). *A commitment scheme* COM *has statistical biding, if for any unbounded adversary $\mathcal{A}$, the advantage*

$$\mathsf{Adv}^{biding}_{\mathsf{COM},\mathcal{A}}(\lambda) := \left| \Pr \left[\begin{array}{c} ck \leftarrow \mathsf{Gen}(1^\lambda) \\ (c, m_0, m_1, d_0, d_1) \leftarrow \mathcal{A}(ck) \end{array} : \begin{array}{c} m_0 \neq m_1 \\ \wedge \ \mathsf{Ver}(ck, c, m_0, d_0) = 1 \\ \wedge \ \mathsf{Ver}(ck, c, m_1, d_1) = 1 \end{array} \right] \right|$$

is negligible over λ.

Definition 3 (Hiding of Commitments). *A commitment scheme* COM *has hiding, if for any PPT adversary $\mathcal{A}$, the advantage*

$$\mathsf{Adv}^{hiding}_{\mathsf{COM},\mathcal{A}}(\lambda) := \left| \Pr \left[\begin{array}{c} ck \leftarrow \mathsf{Gen}(1^\lambda); (m_0, m_1, st) \leftarrow \mathcal{A}(ck) \\ (c, d) \leftarrow \mathsf{Com}(ck, m_0) \end{array} : \mathcal{A}(st, c) = 1 \right] \right.$$
$$\left. - \Pr \left[\begin{array}{c} ck \leftarrow \mathsf{Gen}(1^\lambda); (m_0, m_1, st) \leftarrow \mathcal{A}(ck) \\ (c, d) \leftarrow \mathsf{Com}(ck, m_1) \end{array} : \mathcal{A}(st, c) = 1 \right] \right|$$

is negligible over λ.

Bit commitment schemes (i.e., the message is one bit) with statistical biding can be constructed from one-way functions [14].

2.2 Trapdoor Commitments

Definition 4 (Trapdoor Commitments with Specific Adaptable Message). *Let $\mathcal{M}$ be a message space and $m_{\mathbf{0}} \in \mathcal{M}$. A trapdoor commitment (TC) scheme with specific adaptable message $m_{\mathbf{0}}$ consists of the following four algorithms. Namely,* $\mathsf{TC} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Ver}, \mathsf{TdOpen})$.

- $(ck, td) \leftarrow \mathsf{Gen}(1^\lambda)$. *The key generation algorithm takes as input the security parameter λ, and outputs a commitment key ck and a trapdoor td.*
  We implicitly assume ck is contained in td, and there exists an efficient function to check the validity of a trapdoor w.r.t. a commitment key, i.e., $f_{\mathsf{TC}}(ck, td) = 1$ if td is valid w.r.t. ck.
- $(c, d) \leftarrow \mathsf{Com}(ck, m)$. *The commitment algorithm takes as input ck and a message $m \in \mathcal{M}$, and outputs a commitment c and an opening d.*
- $0/1 \leftarrow \mathsf{Ver}(ck, c, m, d)$. *The verification algorithm takes as input ck, c, m and d, and outputs a bit.*

– $d \leftarrow \mathsf{TdOpen}(td, c, m_{\mathbf{0}}, d_{\mathbf{0}}, m)$. *The trapdoor open algorithm takes as input* td*,* c*,* $m_{\mathbf{0}}$*,* $d_{\mathbf{0}}$*, and another message* m*, and outputs an adapted opening* d*.*

Correctness. *For any* $(ck, td) \leftarrow \mathsf{Gen}(1^{\lambda})$*, any message* $m \in \mathcal{M}$*, the followings two hold.*

1. *If* $(c, d) \leftarrow \mathsf{Com}(ck, m)$*, then* $\mathsf{Ver}(ck, c, m, d) = 1$.
2. *If* $(c, d_{\mathbf{0}}) \leftarrow \mathsf{Com}(ck, m_{\mathbf{0}})$ *and* $d \leftarrow \mathsf{TdOpen}(td, c, m_{\mathbf{0}}, d_{\mathbf{0}}, m)$*, then* $\mathsf{Ver}(ck, c, m, d) = 1$.

Definition 5 (Hiding of Trapdoor Commitments). *A trapdoor commitment scheme* TC *with specific adaptable message* $m_{\mathbf{0}}$ *has hiding, if for any PPT adversary* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{hiding}_{\mathsf{TC},\mathcal{A}}(\lambda) := \left| \Pr\left[\begin{matrix}(ck, td) \leftarrow \mathsf{Gen}(1^{\lambda}); (m, st) \leftarrow \mathcal{A}(ck) \\ (c, d_{\mathbf{0}}) \leftarrow \mathsf{Com}(ck, m_{\mathbf{0}})\end{matrix} : \mathcal{A}(st, c) = 1\right] - \Pr\left[\begin{matrix}(ck, td) \leftarrow \mathsf{Gen}(1^{\lambda}); (m, st) \leftarrow \mathcal{A}(ck) \\ (c, d) \leftarrow \mathsf{Com}(ck, m)\end{matrix} : \mathcal{A}(st, c) = 1\right] \right|$$

is negligible over λ*.*

Definition 6 (Trapdoor Extractability of Trapdoor Commitments). *A trapdoor commitment scheme* TC *with specific adaptable message* $m_{\mathbf{0}}$ *is trapdoor extractable, if there is an efficient extract algorithm* Ext *that can extract a trapdoor from a collision with high probability. More precisely, for* $(ck, td) \leftarrow \mathsf{Gen}(1^{\lambda})$*, any* $(c, m_{\mathbf{0}}, d_{\mathbf{0}})$ *and* (c, m, d) *s.t.* $m \neq m_{\mathbf{0}}$ *and* $\mathsf{Ver}(ck, c, m_{\mathbf{0}}, d_{\mathbf{0}}) = \mathsf{Ver}(ck, c, m, d) = 1$*, it holds that*

$$\Pr[f_{\mathsf{TC}}(ck, \mathsf{Ext}(ck, c, m_{\mathbf{0}}, d_{\mathbf{0}}, m, d)) = 0] \leq negl(\lambda),$$

where the probability is taken over the random choice of key generation.

Definition 7 (Adaption Indistinguishability of Trapdoor Commitments). *A trapdoor commitment scheme* TC *with a specific adaptable message* $m_{\mathbf{0}}$ *has adaption indistinguishability, if for any* m*, any PPT adversary* $\mathcal{A}$*, the advantage*

$$\mathsf{Adv}^{aind}_{\mathsf{TC},\mathcal{A}}(\lambda) := \left| \Pr\left[\begin{matrix}(ck, td) \leftarrow \mathsf{Gen}(1^{\lambda}) \\ (c, d_{\mathbf{0}}) \leftarrow \mathsf{Com}(ck, m_{\mathbf{0}}) \\ d \leftarrow \mathsf{TdOpen}(td, c, m_{\mathbf{0}}, d_{\mathbf{0}}, m)\end{matrix} : \mathcal{A}(ck, m_{\mathbf{0}}, m, c, d) = 1\right] - \Pr\left[\begin{matrix}(ck, td) \leftarrow \mathsf{Gen}(1^{\lambda}) \\ (c, d) \leftarrow \mathsf{Com}(ck, m)\end{matrix} : \mathcal{A}(ck, m_{\mathbf{0}}, m, c, d) = 1\right] \right|$$

is negligible over λ*.*

Remark 1 (Classical Trapdoor Commitments). If $m_{\mathbf{0}}$ in the above definitions is replaced with an arbitrary message m, then we define the classical trapdoor commitment schemes, and the corresponding properties of hiding, trapdoor extractability, and adaption indistinguishability. Jumping ahead, the commitment c of the specific adaptable message $m_{\mathbf{0}}$ serves as one part of the message to be signed in the pre-sign process of adaptor signatures. With the knowledge of the witness y, which is the trapdoor in the commitment scheme, one can open this commitment c to the real message to be signed and hence form a valid adapted signature.

2.3 Signatures

Definition 8 (Signatures). *A signature scheme consists of the following three algorithms. Namely,* $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$.

- $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$. *The key generation algorithm takes as input the security parameter λ, and outputs a public key pk and a secret key sk.*
- $\sigma \leftarrow \mathsf{Sign}(sk, m)$. *The signing algorithm takes as input sk and a message m, and outputs a signature σ.*
- $0/1 \leftarrow \mathsf{Ver}(pk, m, \sigma)$. *The verification algorithm takes as input pk, m, and σ, and outputs a bit b indicating the validity of σ (w.r.t. m).*

Correctness. *For any $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, any message m and $\sigma \leftarrow \mathsf{Sign}(sk, m)$, it holds that $\mathsf{Ver}(pk, m, \sigma) = 1$.*

Definition 9 (Unforgeability of Signatures). *A signature scheme* SIG *is unforgeable under chosen message attacks (UF-CMA secure), if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{uf}_{\mathsf{SIG},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{uf}_{\mathsf{SIG},\mathcal{A}}(\lambda) \Rightarrow 1]$ is negligible over λ, where $\mathsf{Exp}^{uf}_{\mathsf{SIG},\mathcal{A}}(\lambda)$ is defined in Fig. 3.*

```
$\mathsf{Exp}^{uf}_{\mathsf{SIG},\mathcal{A}}(\lambda)$:
$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$; $\mathcal{S} := \varnothing$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}(\cdot)}(pk)$

Return $((m^* \notin \mathcal{S}) \wedge (\mathsf{Ver}(pk, m^*, \sigma^*)))$

$\textsc{Sign}(m)$:
$\sigma \leftarrow \mathsf{Sign}(sk, m)$
$\mathcal{S} := \mathcal{S} \cup \{m\}$
Return $\sigma$
```

Fig. 3. The UF-CMA security experiment of signatures

2.4 NP Languages

Let $\{R_\lambda\} \subseteq (\{0,1\}^* \times \{0,1\}^*)_\lambda$ be a series of binary relations indexed by parameter λ. If λ is fixed then we simply denote R_λ as R. We call R an NP relation if there is an efficient algorithm to check whether $(Y, y) \in R$. The relation R defines an NP language $\mathcal{L}_R := \{Y \in \{0,1\}^* \mid \exists y \in \{0,1\}^* \ s.t. \ (Y, y) \in R\}$. We call Y the instance (not necessarily in $\mathcal{L}_R$), and y a witness of Y if $(Y, y) \in R$. Usually, there is an efficient sample algorithm that returns an instance-witness pair. Formally, $(Y, y) \leftarrow \mathsf{Sample}(R)$.

Definition 10 (Hard Relations). *A binary relation R is hard (one-way) if for any PPT adversary $\mathcal{A}$, its advantage*

$$\mathsf{Adv}^{ow}_{R,\mathcal{A}}(\lambda) := \Pr[(Y, y) \leftarrow \mathsf{Sample}(R); y' \leftarrow \mathcal{A}(R, Y) \ : \ (Y, y') \in R]$$

is negligible over λ.

3 Adaptor Signatures

In this section, we present the definition of adaptor signatures and the security requirements, including the newly proposed witness hiding property.

Definition 11 (Adaptor Signatures). *An adaptor signature scheme w.r.t. a relation R consists of seven algorithms $\mathsf{AS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver}, \mathsf{pSign}, \mathsf{pVer}, \mathsf{Adapt}, \mathsf{Ext})$, where the first three algorithms are defined as regular signatures (cf. Def. 8), and the last four are defined as follows.*

- *$\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$. The pre-sign algorithm takes as input sk, m and an instance Y, and outputs a pre-signature $\tilde{\sigma}$.*
- *$b \leftarrow \mathsf{pVer}(pk, m, Y, \tilde{\sigma})$. The pre-verification algorithm takes as input pk, m, Y and $\tilde{\sigma}$, and outputs a bit indicating the validity of $\tilde{\sigma}$.*
- *$\sigma \leftarrow \mathsf{Adapt}(pk, m, \tilde{\sigma}, y)$. The adaption algorithm takes as input pk, m, $\tilde{\sigma}$ and a witness y as input, and outputs an adapted signature σ.*
- *$y/\perp \leftarrow \mathsf{Ext}(pk, m, Y, \tilde{\sigma}, \sigma)$. The extraction algorithm takes as input pk, m, Y, $\tilde{\sigma}$ and σ, and outputs a witness y, or a failure symbol $\perp$.*

Except the correctness as defined in Def. 8, we additionally require the pre-signature correctness and extraction correctness.

***Pre-signature Correctness and Extraction Correctness**. For any $(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$, any message m, any $(Y, y) \in \mathcal{R}$, $\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$, and $\sigma \leftarrow \mathsf{Adapt}(pk, m, \tilde{\sigma}, y)$, it holds that*

1. *(Pre-signature correctness) $\mathsf{pVer}(pk, m, Y, \tilde{\sigma}) = 1$, and*
2. *(Extraction correctness) $\mathsf{Ver}(pk, m, \sigma) = 1$.*

We require unforgeability, witness extractability and pre-signature adaptability for the security of adaptor signatures. Here, we mainly follow the definition by Dai *et al.* [10] which allows multiple queries to the pre-sign oracle. Meanwhile, we divide the full extractability in [10] into unforgeability and witness extractability (as [2]) for better presenting the different security aspects of adaptor signatures.

Definition 12 (Unforgeability of Adaptor Signatures). *An adaptor signature scheme AS w.r.t. binary relation R is unforgeable under chosen message attacks (UF-CMA secure), if for any PPT adversary $\mathcal{A}$, $\mathsf{Adv}^{uf}_{\mathsf{AS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{uf}_{\mathsf{AS},\mathcal{A}}(\lambda) \Rightarrow 1]$ is negligible over λ, where $\mathsf{Exp}^{uf}_{\mathsf{AS},\mathcal{A}}(\lambda)$ is defined in Fig. 4.*

```
$\mathsf{Exp}^{uf}_{\mathsf{AS},\mathcal{A}}(\lambda)$:
$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$; $\mathcal{S} := \varnothing$; $\mathcal{T}[m] := \varnothing$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}(\cdot),\textsc{pSign}(\cdot,\cdot),\textsc{NewY}()}(pk)$

Return 1 if $(b_1 \wedge (b_{2,1} \vee b_{2,2}))$, and 0 otherwise, where
$b_1$: $\mathsf{Ver}(pk, m^*, \sigma^*) = 1 \ \wedge \ m^* \notin \mathcal{S}$
$b_{2,1}$: $\mathcal{T}[m^*] = \varnothing$
$b_{2,2}$: $\forall (Y, \tilde{\sigma}) \in \mathcal{T}[m^*] : Y \in \mathcal{Y}$
```

```
$\textsc{Sign}(m)$:
$\sigma \leftarrow \mathsf{Sign}(sk, m)$
$\mathcal{S} := \mathcal{S} \cup \{m\}$
Return $\sigma$

$\textsc{pSign}(m, Y)$:
$\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
$\mathcal{T}[m] := \mathcal{T} \cup \{(Y, \tilde{\sigma})\}$
Return $\tilde{\sigma}$

$\textsc{NewY}()$:
$(Y, y) \leftarrow \mathsf{Sample}(R)$
$\mathcal{Y} := \mathcal{Y} \cup \{Y\}$
Return $Y$
```

Fig. 4. The UF-CMA security experiment of adaptor signatures

```
$\mathsf{Exp}^{we}_{\mathsf{AS},\mathcal{A}}(\lambda)$:
$(pk, sk) \leftarrow \mathsf{Gen}(1^\lambda)$; $\mathcal{S} := \varnothing$; $\mathcal{T}[m] := \varnothing$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}(\cdot),\textsc{pSign}(\cdot,\cdot)}(pk)$

Return 1 if $(b_1 \wedge b_2)$, and 0 otherwise, where
$b_1$: $\mathsf{Ver}(pk, m^*, \sigma^*) = 1 \ \wedge \ m^* \notin \mathcal{S}$
$b_2$: $\forall (Y, \tilde{\sigma}) \in \mathcal{T}[m^*] : (Y, \mathsf{Ext}(pk, m^*, Y, \tilde{\sigma}, \sigma^*)) \notin R$
// all $(Y, \tilde{\sigma})$ in the pre-sign list lead to a failed extraction
```

```
$\textsc{Sign}(m)$:
$\sigma \leftarrow \mathsf{Sign}(sk, m)$
$\mathcal{S} := \mathcal{S} \cup \{m\}$
Return $\sigma$

$\textsc{pSign}(m, Y)$:
$\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$
$\mathcal{T}[m] := \mathcal{T} \cup \{(Y, \tilde{\sigma})\}$
Return $\tilde{\sigma}$
```

Fig. 5. The witness extractability experiment of adaptor signatures

Definition 13 (Witness Extractability). *An adaptor signature scheme* AS *w.r.t. relation* R *is witness extractable, if for any PPT adversary* $\mathcal{A}$, $\mathsf{Adv}^{we}_{\mathsf{AS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{we}_{\mathsf{AS},\mathcal{A}}(\lambda) \Rightarrow 1]$ *is negligible over* λ, *where* $\mathsf{Exp}^{we}_{\mathsf{AS},\mathcal{A}}(\lambda)$ *is defined in Fig. 5.*

Remark 2 (Stronger Definition of Witness Extractability). A stronger definition of witness extractability is that the extraction will not fail as long as $\mathsf{pVer}(pk, m, Y, \tilde{\sigma}) = 1$. Combined with the correctness, the difference between this stronger definition and Def. 13 lies in whether one can extract a witness from a valid signature$\tilde{\sigma}$ and an adapted signature σ, such that $\tilde{\sigma}$ is not generated via pSign.

However, such a stronger definition is not practical in the real world, since witness extractability is meant to guarantee Alice's right to extract y once Bob publishes the adapted signature, assuming Alice generates the pre-signature via (normal) pSign, but not other ways.

Definition 14 (Pre-signature Adaptability). *An adaptor signature scheme* AS *w.r.t. relation* R *has pre-signature adaptability, if for any public key* pk, *any* *message* m*, any* $(Y, y) \in R$ *and pre-signature* $\tilde{\sigma}$ *s.t.* $\mathsf{pVer}(pk, m, Y, \tilde{\sigma}) = 1$*, it holds that* $\mathsf{Ver}(pk, m, \mathsf{Adapt}(pk, m, \tilde{\sigma}, y)) = 1$*.*

Now we formally define the property that y can be extracted from both the pre-signature $\tilde{\sigma}$ and the adapted signature σ, but not just σ. In other words, σ leaks no additional information about y.

Definition 15 (Witness Hiding of Adaptor Signatures). *An adaptor signature scheme* AS *w.r.t. relation* R *is witness hiding, if there exists a simulator* Sim *such that, for any PPT adversary* $\mathcal{A}$*,*

$$\mathsf{Adv}^{wh}_{\mathsf{AS},\mathsf{Sim}\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{wh}_{\mathsf{AS},\mathsf{Sim},\mathcal{A},0}(\lambda) \Rightarrow 1] - \Pr[\mathsf{Exp}^{wh}_{\mathsf{AS},\mathsf{Sim},\mathcal{A},1}(\lambda) \Rightarrow 1]|$$

is negligible over λ*, where* $\mathsf{Exp}^{wh}_{\mathsf{AS},\mathsf{Sim},\mathcal{A},b}(\lambda)$ *(*$b \in \{0, 1\}$*) are defined in Fig. 6.*

$\mathsf{Exp}^{wh}_{\mathsf{AS},\mathsf{Sim},\mathcal{A},b}(\lambda)$:	$\textsc{Chall}_0(pk, sk, m)$	$\textsc{Chall}_1(pk, sk, m)$
$(Y, y) \leftarrow \mathsf{Sample}(R)$	If $f_{\mathsf{AS}}(pk, sk) \neq 1$: Return $\perp$	If $f_{\mathsf{AS}}(pk, sk) \neq 1$: Return $\perp$
Return $\mathcal{A}^{\textsc{Chall}_b(\cdot,\cdot,\cdot)}(Y)$	// check the validity of (pk, sk)	// check the validity of (pk, sk)
	$\tilde{\sigma} \leftarrow \mathsf{pSign}(sk, m, Y)$	$\sigma \leftarrow \mathsf{Sim}(pk, sk, m, Y)$
	$\sigma \leftarrow \mathsf{Adapt}(pk, m, \tilde{\sigma}, y)$	Return σ
	Return σ	

Fig. 6. The witness hiding experiments of adaptor signatures

Remark 3 (On the Formalization of Witness Hiding). Witness hiding property requires that the witness is exposed from both the pre-signature $\tilde{\sigma}$ and the adapted signature σ, but not from either of them. One might wonder why we only ask the adapted signature σ to leak no information about the witness in the above definition, but do not impose restrictions on the pre-signature $\tilde{\sigma}$. Actually, the witness hiding property for $\tilde{\sigma}$ is naturally established, since the pre-sign algorithm takes only the secret key sk, the message m, and the instance Y as input, and hence it is independent of y.

The motivation of introducingwitness hiding is to prevent a malicious eavesdropper who has access to blockchains from extracting the witness using an adapted signature. From this point, the eavesdropper $\mathcal{A}_{eav}$ has no knowledge of the (signing) secret key, and it should be unable to distinguish a pre-sign-and-adapt signature (using y) and a simulated signature. This can be formalized by an experiment where $\mathcal{A}_{eav}$ has access to oracles $\textsc{Chall}_0$ (the pre-sign-and-adaption oracle) and $\textsc{Chall}_1$ (the simulation oracle), and the secret key is sampled by the challenger/experiment and not known to $\mathcal{A}_{eav}$. In the definition above, we consider a stronger adversary $\mathcal{A}$ who is able to select the signing secret key by itself, and $\mathcal{A}$ degrades into an eavesdropper adversary $\mathcal{A}_{eav}$ if it generates sk honestly and discards it immediately after the query. Therefore, Definition 15 is stronger than the definition which considers just the eavesdropping case.

Remark 4 (On the Relationship with Unlinkability [10]*).* The unlinkability property, as defined in [10], requires that a signature obtained by first pre-signing and then adapting is indistinguishable from a signature obtained by directly signing the message. Unlinkability implies witness hiding, as the simulator Sim can be replaced by the signing algorithm. However, witness hiding does not imply unlinkability. To see this, consider a witness hiding adaptor signature scheme AS with a simulator Sim. If we modify the pre-sign algorithm so that it signs $(m||0)$ instead of m, and adjust the verification and adaptation algorithms accordingly, then the scheme still satisfies witness hiding. However, the unlinkability property does not hold in this modified scheme, since the pre-sign-then-adapt mode returns a signature for $(m||0)$, while the direct sign mode returns a signature for m.

Though unlinkability [10] is strictly stronger than the witness hiding property defined in this work, it is still reasonable to introduce the *weaker* definition of witness hiding.

- In some applications, we may want to make a pre-sign-and-adapt signature be *distinguishable* from a directly signed signature (while hiding the witness from just the adapted signature at the same time). For example, Alice may use the same signing secret key in both a blockchain transaction system (where Alice pre-signs transactions only) and a daily authentication system (where Alice signs messages directly only), and she wants to distinguish all signatures in the blockchain systems so that she can trace and analyze her behaviors. The unlinkability property is overengineered in this scenario.
- It is essential to explore the minimal requirement for practical AS where the witness is hidden from just the adapted signatures. As shown in this work, such *weaker* definition allows us to design witness hiding AS for all NP relations.
- The unlinkability defined in [10] is very strong, since the adversary is able to select (Y, y) by itself. To achieve unlinkability, [10] requires the strong random-self reducibility of the underlying NP relations, and therefore the instantiations are limited to number theoretical problems (DL, LWE, etc.). It is very challenging to design unlinked AS schemes for arbitrary relations, where the pre-sign-and-adapt signature (in which the instance is specified by the adversary, i.e., $\mathsf{PreSignAdapt}(sk, m, Y, y)$) is indistinguishable from a normal one (in which there is no instance as input at all, i.e., $\mathsf{Sign}(sk, m)$). This means that the instance Y can be re-randomized or eliminated during the adaption, which seems inherently requires some special structure of the underlying NP relation. It is unknown whether unlinked AS schemes for any NP relations exist.

4 Generic Construction of Adaptor Signatures from Signatures and Trapdoor Commitments with a Specific Adaptable Message

Let $\mathcal{M}$ be a message space, and there is a fixed message $m_{\mathbf{0}} \in \mathcal{M}$ (e.g., the all-zero bit string). Let $\mathsf{SIG} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ be a signature scheme with the message space $\mathcal{M}$. Let R be an NP relation, and any $(Y, y) \in \mathcal{R}$ forms a trapdoor commitment scheme $\mathsf{TC} = (\mathsf{Gen}, \mathsf{Com}, \mathsf{Ver}, \mathsf{TdOpen})$ with a specific adaptable message $m_{\mathbf{0}}$, where Y is the commitment key and y is the trapdoor, and $\mathsf{TC}.\mathsf{Gen}(1^\lambda)$ just returns $(Y, y) \leftarrow \mathsf{Sample}(R)$.

Our adaptor signature scheme AS with the message space $\mathcal{M} \setminus \{m_{\mathbf{0}}\}$ is shown in Fig. 7.

```
$\underline{\mathsf{Gen}(1^\lambda):}$
$(pk, sk) \leftarrow \mathsf{SIG.Gen}(1^\lambda)$
Return $(pk, sk)$

$\underline{\mathsf{Sign}(sk, m):}$
$(Y, y) \leftarrow \mathsf{Sample}(R)$
$(c, d) \leftarrow \mathsf{TC.Com}(Y, m)$
$\bar{\sigma} \leftarrow \mathsf{SIG.Sign}(sk, (m, Y, c))$
Return $\sigma := (\bar{\sigma}, Y, c, d)$

$\underline{\mathsf{Ver}(pk, m, \sigma):}$
Parse $\sigma = (\bar{\sigma}, Y, c, d)$
If $\mathsf{TC.Ver}(Y, c, m, d) = 0$: return 0
Return $\mathsf{SIG.Ver}(pk, (m, Y, c), \bar{\sigma})$
```

```
$\underline{\mathsf{pSign}(sk, m, Y):}$
$(c, d_{\mathbf{0}}) \leftarrow \mathsf{TC.Com}(Y, m_{\mathbf{0}})$
$\bar{\sigma} \leftarrow \mathsf{SIG.Sign}(sk, (m, Y, c))$
Return $\tilde{\sigma} := (\bar{\sigma}, Y, c, d_{\mathbf{0}})$

$\underline{\mathsf{pVer}(pk, m, Y, \tilde{\sigma}):}$
Parse $\tilde{\sigma} = (\bar{\sigma}, Y, c, d_{\mathbf{0}})$
If $\mathsf{TC.Ver}(Y, c, m_{\mathbf{0}}, d_{\mathbf{0}}) = 0$: return 0
Return $\mathsf{SIG.Ver}(pk, (m, Y, c), \bar{\sigma})$

$\underline{\mathsf{Adapt}(pk, m, \tilde{\sigma}, y):}$
Parse $\tilde{\sigma} = (\bar{\sigma}, Y, c, d_{\mathbf{0}})$
$d \leftarrow \mathsf{TC.TdOpen}(y, c, m_{\mathbf{0}}, d_{\mathbf{0}}, m)$
Return $\sigma := (\bar{\sigma}, Y, c, d)$

$\underline{\mathsf{Ext}(pk, m, Y, \tilde{\sigma}, \sigma):}$
Parse $\tilde{\sigma} = (\bar{\sigma}', Y', c', d_{\mathbf{0}})$ and $\sigma = (\bar{\sigma}, Y, c, d)$
If $(\bar{\sigma}' \neq \bar{\sigma}) \vee (Y' \neq Y) \vee (c' \neq c)$: return $\perp$
Return $\mathsf{TC.Ext}(Y, c, m_{\mathbf{0}}, d_{\mathbf{0}}, m, d)$
```

Fig. 7. Generic construction of adaptor signatures from signatures and trapdoor commitments with a specific adaptable message $m_{\mathbf{0}}$

Correctness. The correctness of AS consists of three aspects.

- Signature correctness & Pre-signature correctness. These are guaranteed by the correctness of SIG and the correctness of TC (the first property of TC).
- Extraction correctness. This is guaranteed by the trapdoor extractability of TC.

Theorem 2. *If* SIG *has UF-CMA security,* TC *has hiding and trapdoor extractability and R is a hard relation, then the adaptor signature scheme* AS *in Fig. 7 has UF-CMA security, witness extractability, pre-signature adaptability and witness hiding.*

Proof. **UF-CMA security**. Let $(m^*, \sigma^* = (\bar{\sigma}^*, Y^*, c^*, d^*))$ be $\mathcal{A}$'s final forgery in the unforgeability experiment (cf. Def. 12), and $m^* \in \mathcal{M} \setminus \{m_\mathbf{0}\}$. Recall that for $\mathcal{A}$ to win, it must hold that

1. $\mathsf{SIG.Ver}(pk, (m^*, Y^*, c^*), \bar{\sigma}^*) = 1$ and $\mathsf{TC.Ver}(Y^*, c^*, m^*, d^*) = 1$, and
2. $\mathcal{A}$ never queries $\textsc{Sign}(m^*)$, and
3. (a) either $\mathcal{T}[m^*] = \varnothing$ (i.e., $\mathcal{A}$ never asks $\textsc{pSign}(m^*, Y)$ for any Y), or
   (b) for all $(Y, \tilde{\sigma} = (\bar{\sigma}, Y, c, d)) \in \mathcal{T}[m^*]$, it holds that $Y \in \mathcal{Y}$ (i.e., $\mathcal{A}$ only queries $\textsc{pSign}(m^*, Y)$ for Y whose witness is unknown to it).

We first analyze the case $1 \wedge 2 \wedge (a)$. It is easy to see that in this case, the challenger $\mathcal{C}$ does not sign a message of the form $(m^*, \cdot, \cdot)$ when answering the pre-signing oracle $\textsc{pSign}$ and the signing oracle $\textsc{Sign}$. Therefore, we can easily construct a reduction algorithm to break the UF-CMA security of the underlying SIG.

Then we analyze the case $1 \wedge 2 \wedge (b)$. We divide it into the following two subcases.

(i) For all $(Y, \tilde{\sigma} = (\bar{\sigma}, Y, c, d)) \in \mathcal{T}[m^*]$, $(Y, c) \neq (Y^*, c^*)$.
Similarly, this means that $\mathcal{C}$ does not sign a message in the form of (m^*, Y^*, c^*) when answering the pre-signing oracle $\textsc{pSign}$. Therefore, $\mathcal{A}$ breaks the UF-CMA security of the underlying SIG.

(ii) There exists $(Y, \tilde{\sigma} = (\bar{\sigma}, Y, c, d_\mathbf{0})) \in \mathcal{T}[m^*]$ such that $(Y, c) = (Y^*, c^*)$.
Recall that the message space defined in AS is $\mathcal{M} \setminus \{m_\mathbf{0}\}$. Therefore $m^* \neq m_\mathbf{0}$. Besides, we have $\mathsf{TC.Ver}(Y, c, m_\mathbf{0}, d_\mathbf{0}) = 1$ due to the correctness of AS. Combined with the fact that $\mathsf{TC.Ver}(Y^*, c^*, m^*, d^*) = 1$ and $(Y, c) = (Y^*, c^*)$, we can extract a witness y' via $\mathsf{TC.Ext}(Y^*, c^*, m_\mathbf{0}, d_\mathbf{0}, m^*, d^*)$. Since $Y^* \in \mathcal{Y}$ (i.e., the corresponding witness of Y^* is unknown to $\mathcal{A}$), $\mathcal{A}$ breaks the one-wayness of the hard relation R.

The UF-CMA security holds as a result.

Witness Extractability. Let (m^*, σ^*) be $\mathcal{A}$'s final output in the witness extractability experiment (cf. Def. 13), and $\sigma^* = (\bar{\sigma}^*, Y^*, c^*, d^*)$. Recall that for $\mathcal{A}$ to win, we have

1. $\mathsf{SIG.Ver}(pk, (m^*, Y^*, c^*), \bar{\sigma}^*) = 1$ and $\mathsf{TC.Ver}(Y^*, c^*, m^*, d^*) = 1$, and $\mathcal{A}$ never asks $\textsc{Sign}(m^*)$.
2. For all $(Y, \tilde{\sigma} = (\bar{\sigma}, Y, c, d_\mathbf{0})) \in \mathcal{T}[m^*]$, $(Y^*, \mathsf{TC.Ext}(Y, c, m_\mathbf{0}, d_\mathbf{0}, m^*, d^*)) \notin R$ (i.e., the witness extraction fails for all $(Y, \tilde{\sigma}) \in \mathcal{T}[m^*]$).

Since $\mathcal{A}$ never queries $\textsc{Sign}(m^*)$, and $\bar{\sigma}^*$ is a valid signature w.r.t. (m^*, Y^*, c^*), there must exist an item $(Y, \tilde{\sigma} = (\bar{\sigma}, Y, c, d_\mathbf{0})) \in \mathcal{T}[m^*]$ s.t., $Y = Y^*$ and $c = c^*$, as otherwise $\mathcal{A}$ would break the UF-CMA security of the underlying SIG scheme. Then, for that $(Y, \tilde{\sigma}) \in \mathcal{T}[m^*]$, we have $\mathsf{SIG.Ver}(pk, (m^*, Y, c), \bar{\sigma}) = 1$ and $\mathsf{TC.Ver}(Y, c, m_\mathbf{0}, d_\mathbf{0}) = 1$ due to the correctness of AS. Therefore, from the fact that $\mathsf{TC.Ver}(Y^*, c^*, m^*, d^*) = 1$, $\mathsf{TC.Ver}(Y = Y^*, c = c^*, m_\mathbf{0}, d_\mathbf{0}) = 1$ and $m^* \neq m_\mathbf{0}$, the extraction algorithm $\mathsf{TC.Ext}(Y^*, c^*, m_\mathbf{0}, d_\mathbf{0}, m^*, d^*)$ will always return a witness y satisfying $(Y^*, y) \in R$, which completes the proof of witness extractability.

Pre-signature Adaptability. Let $\tilde{\sigma} = (\bar{\sigma}, Y, c, d_{\mathbf{0}})$ be a pre-signature such that $\mathsf{pVer}(pk, m, Y, \tilde{\sigma}) = 1$. Namely, $\mathsf{SIG.Ver}(pk, (m, Y, c), \bar{\sigma}) = 1$ and $\mathsf{TC.Ver}(Y, c, m_{\mathbf{0}}, d_{\mathbf{0}}) = 1$. Assume $(Y, y) \in R$, the adaption algorithm will return $\sigma = (\bar{\sigma}, Y, c, d)$ where $d \leftarrow \mathsf{TC.TdOpen}(y, c, m_{\mathbf{0}}, d_{\mathbf{0}}, m)$. Obviously $\mathsf{SIG.Ver}(pk, (m, Y, c), \bar{\sigma}) = 1$ still holds. Furthermore, due to the correctness of TC, we have $\mathsf{TC.Ver}(Y, c, m, d) = 1$, and the pre-signature adaptability holds consequently.

Witness Hiding. To prove the witness hiding property, we have to design a simulator Sim such that it can simulate an adapted signature given (m, Y), which is indistinguishable from a signature generated by the pre-sign-and-adaption paradigm with the knowledge of y. (Recall that the witness hiding property assumes that the secret key sk for signing is also given to Sim as an input.)

We design Sim similarly to the signing algorithm Sign, with the only difference being that Sim uses a fixed Y instead of sampling a new Y.

Then we argue that a simulated signature $\sigma_1 = (\bar{\sigma}_1, Y, c_1, d_1)$ is indistinguishable from a pre-sign-and-adapt signature $\sigma_0 = (\bar{\sigma}_0, Y, c_0, d_0)$. Notice that $\bar{\sigma}_1 \leftarrow \mathsf{SIG.Sign}(sk, (m, Y, c_1))$ and $\bar{\sigma}_0 \leftarrow \mathsf{SIG.Sign}(sk, (m, Y, c_0))$. Therefore, it is sufficient to prove that (c_1, d_1) is indistinguishable from (c_0, d_0).

- In the simulated signature, (c_1, d_1) is computed via $(c_1, d_1) \leftarrow \mathsf{TC.Com}(Y, m)$.
- In the pre-sign-and-adaption signature, (c_0, d_0) is computed via $(c_0, d_{\mathbf{0}}) \leftarrow \mathsf{Com}(Y, m_{\mathbf{0}})$ and $d_0 \leftarrow \mathsf{TC.TdOpen}(y, c_0, m_{\mathbf{0}}, d_{\mathbf{0}}, m)$.

According to the adaption indistinguishability of TC, (c_1, d_1) and (c_0, d_0) are indistinguishable, and the witness hiding of AS holds consequently. $\square$

5 Trapdoor Commitments for Any NP Relation

In this section we show a trapdoor commitment with a specific adaptable message for the Hamiltonian cycle problem, a well-known NP complete problem. Combined with Theorem 2, we obtain adaptor signatures for any NP relation.

Zero-Knowledge Proof for the Hamiltonian Cycle Problem. Let us first recall the zero-knowledge protocol for the Hamiltonian cycle problems by Blum [7,14]. Let G be a graph, and $H \subseteq G$ be a Hamiltonian cycle, i.e., a witness of Hamiltonian graph instance G. The zero-knowledge protocol between the prover $\mathcal{P}$ and the verifier $\mathcal{V}$ is shown as follows.

1. $\mathcal{P}$ randomly samples a permutation π and gets $G' := \pi(G)$. Then $\mathcal{P}$ commits the adjacency matrix of G' and sends the commitments $com_{G'}$ to $\mathcal{V}$.
2. After receiving $com_{G'}$, $\mathcal{V}$ sends a random bit $b \xleftarrow{\$} \{0, 1\}$.
3. $\mathcal{P}$ responses as follows.
   - If $b = 0$, then $\mathcal{P}$ sends all openings of $com_{G'}$, and the permutation π to $\mathcal{V}$.

  - If $b = 1$, then $\mathcal{P}$ sends all openings of $com_{H'}$ to $\mathcal{V}$, where $H' := \pi(H)$ is a Hamiltonian cycle of G'.
4. $\mathcal{V}$ checks as follows.
  - If $b = 0$, then $\mathcal{V}$ checks $com_{G'}$ are commitments of the adjacency matrix of $\pi(G)$.
  - If $b = 1$, then $\mathcal{V}$ checks $com_{G'}$ include commitments of the adjacency matrix of H', and H' is a Hamiltonian cycle.

Theorem 3 **([7,14]).** *If the commitment scheme has statistical biding and (computational) hiding, then the above protocol is a zero-knowledge proof protocol with soundness error 1/2.*

Now we show the zero-knowledge proof (Sigma) protocol can be transferred into a bit trapdoor commitment with special adaptable message $m_{\mathbf{0}} = 0$. At a high-level, the transform follows the proof of the equivalence of Sigma protocols and chameleon hashes by Bellare and Ristov [4]. Specifically, the commitment, challenge, and response in a sigma protocol correspond to the commitment, message, and opening in a trapdoor commitment scheme, respectively, with the witness of the Sigma protocol serving as the trapdoor in the trapdoor commitment scheme. In more detail, the bit trapdoor commitment scheme is as follows. Here, $\mathcal{M} = \{0, 1\}$ and $m_{\mathbf{0}} = 0$.

- $\mathsf{Gen}(1^\lambda)$. Randomly sample a Hamiltonian graph G with a Hamiltonian cycle $H \subseteq G$ as its witness. Return $(ck, td) := (G, H)$.
- $\mathsf{Com}(ck, m \in \{0, 1\})$.
  - If $m = 0$, then randomly sample a permutation π, and commit the adjacency matrix of $G' := \pi(G)$ to get $com_{G'}$ and the corresponding openings $d_{G'}$. Return $(c, d) := (com_{G'}, (\pi, d_{G'}))$.
  - If $m = 1$, then randomly generate a Hamiltonian graph G' with a Hamiltonian cycle H', and commit the adjacency matrix of G' to get $com_{G'}$ and the corresponding openings $d_{G'}$. Return $(c, d) := (com_{G'}, (H', d_{H'}))$.
- $\mathsf{Ver}(ck, c, m, d)$.
  - If $m = 0$, parse $(c, d) = (com_{G'}, (\pi, d_{G'}))$. Return 1 if $com_{G'}$ are the commitments of $\pi(G)$, and 0 otherwise.
  - If $m = 1$, parse $(c, d) = (com_{G'}, (H', d_{H'}))$. Return 1 if $com_{G'}$ includes commitments of a Hamiltonian cycle H', and 0 otherwise.
- $\mathsf{TdOpen}(td = H, c, m_{\mathbf{0}} = 0, d_{\mathbf{0}}, m)$.
  - If $m = 0$, return $d_{\mathbf{0}}$ directly.
  - If $m = 1$, parse $(c, d_{\mathbf{0}}) := (com_{G'}, (\pi, d_{G'}))$. Return $\perp$ if $com_{G'}$ are not the commitments of $\pi(G)$. Otherwise, let $H' := \pi(H)$, and $d_{H'} \subseteq d_{G'}$ be the openings w.r.t. the adjacency matrix of H'. Return $d := (H', d_{H'})$.

The correctness is implied by the completeness of the zero-knowledge protocol for the Hamiltonian cycle problem.

Theorem 4. *If the underlying commitment scheme has statistical biding and computational hiding, then the above constructed trapdoor commitment scheme with specific adaptable message* $m_{\mathbf{0}} = 0$ *has (computational) hiding, trapdoor extractability, and adaption indistinguishability.*

Proof. **Hiding**. This follows directly from the hiding property of the underlying commitment scheme. Namely, for any two graphs G_0 and G_1 of the same size (i.e., with the same number of vertices and edges), the commitments com_{G_0} and com_{G_1} are computationally indistinguishable.

Trapdoor Extractability. Let $(c = com_{G'}, m_{\mathbf{0}} = 0, d_{\mathbf{0}} = (\pi, d_{G'}))$ and $(c = com_{G'}, m = 1, d = (H', d_{H'}))$ be two commitment-message-opening tuples. On one hand, $\mathsf{Ver}(ck, c, m_{\mathbf{0}}, d_{\mathbf{0}}) = 1$ implies that $com_{G'}$ is a commitment of graph $\pi(G)$. On the other hand, $\mathsf{Ver}(ck, c, m, d) = 1$ implies that $com_{G'}$ contains a commitment of Hamiltonian cycle H'. Since the commitment scheme is statistically biding, $H' \subseteq \pi(G)$ holds with overwhelming probability. Therefore, $\pi^{-1}(H') \subseteq G$ is a Hamiltonian cycle, which finishes the proof of trapdoor extractability.

Adaption Indistinguishability. The adaption indistinguishability property requires that the distribution (c, d), where $(c, d_{\mathbf{0}}) \leftarrow \mathsf{Com}(ck, m_{\mathbf{0}})$ and $d \leftarrow \mathsf{TdOpen}(td, c, m_{\mathbf{0}}, d_{\mathbf{0}}, m = 1)$, is indistinguishable from the distribution (c, d), where $(c, d) \leftarrow \mathsf{Com}(ck, m = 1)$.

Recall that in both cases (c, d) is in the form of $(com_{G'}, (H', d_{H'}))$.

- In the first case, $G' = \pi(G)$ and $H' = \pi(H)$, where π is a random permutation.
- In the second case, G' is a randomly sampled graph with a Hamiltonian cycle H'.

Since π is a random permutation, $(H', d_{H'})$ distributed identically in both cases. The only difference lies in the parts of $com_{G' \setminus H'}$, which are computationally indistinguishable due to the hiding property of the underlying commitment scheme. □

Extension for Large Message Space . Via the standard hybrid argument, it is easy to extend the message space from $\{0, 1\}$ to $\{0, 1\}^{\ell}$ for any polynomial ℓ, and the specific adaptable message $m_{\mathbf{0}}$ is 0^{ℓ}.

Further Discussion . Though the above-mentioned construction works for any NP relation R, it involves a heavy Karp reduction [17] from R to the Hamiltonian cycle problem. For commonly used relations in cryptography such as the DL relation, the RSA relation, the LWE relation, and the SIS relation, more efficient trapdoor commitments (with a specific adaptable message) can be constructed from the Schnorr identification scheme [24,25], the RSA-based Sigma protocol/chameleon hashes [1], the LWE-based Sigma protocols [5,9] and the SIS-based Sigma protocols [6,21], respectively. Furthermore, in the full version [20], we show a direct construction of trapdoor commitments for any NP relation with random self-reducibility.

6 Conclusion

In this work we introduce witness hiding for adaptor signatures (AS), which requires that the witness y can be extracted from both a pre-signature and an adapted signature, but not from either of them individually. We propose a generic construction of witness-hiding AS from signatures and trapdoor commitments with a specific adaptable message, a weaker version of trapdoor commitments. Based on the Hamiltonian cycle problem, we also propose a trapdoor commitment scheme with a specific adaptable message, where the commitment key is the Hamiltonian problem instance and the trapdoor is the Hamiltonian cycle witness. Therefore, we prove that the existence of one-way functions implies the existence of witness hiding adaptor signatures for any NP relation.

Further Work. One potential approach to circumvent the heavy Karp reduction [17] is to leverage the multi-party computation in the head (MPCitH) paradigm [16]. Namely, let Π be a secure MPC protocol for $f(\cdot)$, where $f(Y, y_1, ..., y_n)$ is a function with public input Y and private inputs y_i from users U_i for $i \in [n]$, and $f(Y, y_1, ..., y_n)$ outputs 1 if and only if $(Y, \bigoplus_{i\in[n]} y_i) \in R$. Taking the MPCitH paradigm, we immediately obtain a Sigma (zero-knowledge) protocol with special soundness, which is closely related to the witness extractability of AS as discussed earlier. However, converting such a Sigma protocol into a trapdoor commitment scheme with a special adaptable message is not straightforward. In the Sigma protocol, a witness is necessary for generating the commitment (the first message in the protocol) to ensure that there exists a valid response for every possible challenge. On the other hand, in a trapdoor commitment scheme, there is no trapdoor (witness) when generating a commitment for a message. In fact, to construct AS, we require a Sigma protocol of which the commitment does not rely on the witness, which cannot be satisfied by the MPCitH paradigm. We leave constructing more efficient AS for NP relations from MPCitH as a further work.

Acknowledgements. We thank all anonymous reviewers for their valuable comments. This work was done while the authors were at Purdue University. Xiangyu Liu and Ioannis Tzannetos were funded by AnalytiXIN and Sunday Group, Inc. Vassilis Zikas was funded in part by NSF grant No. 2055599, AnalytiXIN, and Sunday Group, Inc.

References

1. Ateniese, G., de Medeiros, B.: Identity-based chameleon hash and applications. In: Juels, A. (ed.) FC 2004, Key West, FL, USA, February 9-12, 2004. Revised Papers. Lecture Notes in Computer Science, vol. 3110, pp. 164–180. Springer (2004). https://doi.org/10.1007/978-3-540-27809-2_19, https://doi.org/10.1007/978-3-540-27809-2_19
2. Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Generalized bitcoin-compatible channels. IACR Cryptol. ePrint Arch. p. 476 (2020), https://eprint.iacr.org/2020/476

3. Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Generalized channels from limited blockchain scripts and adaptor signatures. In: ASIACRYPT 2021. vol. 13091, pp. 635–664. Springer (2021). https://doi.org/10.1007/978-3-030-92075-3_22, https://doi.org/10.1007/978-3-030-92075-3_22
4. Bellare, M., Ristov, T.: A characterization of chameleon hash functions and new, efficient designs. J. Cryptol. **27**(4), 799–823 (2014). https://doi.org/10.1007/S00145-013-9155-8, https://doi.org/10.1007/s00145-013-9155-8
5. Benhamouda, F., Krenn, S., Lyubashevsky, V., Pietrzak, K.: Efficient zero-knowledge proofs for commitments from learning with errors over rings. In: Pernul, G., Ryan, P.Y.A., Weippl, E.R. (eds.) ESORICS 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9326, pp. 305–325. Springer (2015). https://doi.org/10.1007/978-3-319-24174-6_16, https://doi.org/10.1007/978-3-319-24174-6_16
6. Beullens, W.: Sigma protocols for mq, PKP and sis, and fishy signature schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III. Lecture Notes in Computer Science, vol. 12107, pp. 183–211. Springer (2020). https://doi.org/10.1007/978-3-030-45727-3_7, https://doi.org/10.1007/978-3-030-45727-3_7
7. Blum, M.: How to prove a theorem so no one else can claim it. In: Proceedings of the International Congress of Mathematicians. vol. 1, p. 2. Citeseer (1986)
8. Bursuc, S., Mauw, S.: Contingent payments from two-party signing and verification for abelian groups. In: CSF 2022, Haifa, Israel, August 7-10, 2022. pp. 195–210. IEEE (2022). https://doi.org/10.1109/CSF54842.2022.9919674, https://doi.org/10.1109/CSF54842.2022.9919674
9. Corrigan-Gibbs, H.: Lattice-based signatures. Presentation Slides (3 2024), https://65610.csail.mit.edu/2024/lec/l14-latsig.pdf, accessed on May 2024
10. Dai, W., Okamoto, T., Yamamoto, G.: Stronger security and generic constructions for adaptor signatures. In: Isobe, T., Sarkar, S. (eds.) INDOCRYPT 2022, Kolkata, India, December 11-14, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13774, pp. 52–77. Springer (2022). https://doi.org/10.1007/978-3-031-22912-1_3, https://doi.org/10.1007/978-3-031-22912-1_3
11. Erwig, A., Faust, S., Hostáková, K., Maitra, M., Riahi, S.: Two-party adaptor signatures from identification schemes. In: Garay, J.A. (ed.) PKC 2021, Virtual Event, May 10-13, 2021, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12710, pp. 451–480. Springer (2021). https://doi.org/10.1007/978-3-030-75245-3_17, https://doi.org/10.1007/978-3-030-75245-3_17
12. Erwig, A., Riahi, S.: Deterministic wallets for adaptor signatures. In: Atluri, V., Pietro, R.D., Jensen, C.D., Meng, W. (eds.) ESORICS 2022, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13555, pp. 487–506. Springer (2022). https://doi.org/10.1007/978-3-031-17146-8_24, https://doi.org/10.1007/978-3-031-17146-8_24
13. Esgin, M.F., Ersoy, O., Erkin, Z.: Post-quantum adaptor signatures and payment channel networks. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) ESORICS 2020, Guildford, UK, September 14-18, 2020, Proceedings, Part II. Lecture Notes in Computer Science, vol. 12309, pp. 378–397. Springer (2020). https://doi.org/10.1007/978-3-030-59013-0_19, https://doi.org/10.1007/978-3-030-59013-0_19
14. Goldreich, O.: The Foundations of Cryptography - Volume 1: Basic Techniques. Cambridge University Press (2001). https://doi.org/10.1017/CBO9780511546891, http://www.wisdom.weizmann.ac.il/%7Eoded/foc-vol1.html

15. Goldreich, O.: The Foundations of Cryptography - Volume 2: Basic Applications. Cambridge University Press (2004). https://doi.org/10.1017/CBO9780511721656, http://www.wisdom.weizmann.ac.il/%7Eoded/foc-vol2.html
16. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge proofs from secure multiparty computation. SIAM J. Comput. **39**(3), 1121–1152 (2009). https://doi.org/10.1137/080725398, https://doi.org/10.1137/080725398
17. Karp, R.M.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA. pp. 85–103. The IBM Research Symposia Series, Plenum Press, New York (1972). https://doi.org/10.1007/978-1-4684-2001-2_9, https://doi.org/10.1007/978-1-4684-2001-2_9
18. Klamti, J.B., Hasan, M.A.: Post-quantum two-party adaptor signature based on coding theory. Cryptogr. **6**(1), 6 (2022). https://doi.org/10.3390/CRYPTOGRAPHY6010006, https://doi.org/10.3390/cryptography6010006
19. Krawczyk, H., Rabin, T.: Chameleon hashing and signatures. IACR Cryptol. ePrint Arch. p. 10 (1998), http://eprint.iacr.org/1998/010
20. Liu, X., Ioannis, T., Zikas, V.: Adaptor signatures: New security definition and A generic construction for NP relations. IACR Cryptol. ePrint Arch. p. 1051 (2024), https://eprint.iacr.org/2024/1051
21. Lyubashevsky, V.: Lattice-based identification schemes secure under active attacks. In: Cramer, R. (ed.) PKC 2008, Barcelona, Spain, March 9-12, 2008. Proceedings. Lecture Notes in Computer Science, vol. 4939, pp. 162–179. Springer (2008). https://doi.org/10.1007/978-3-540-78440-1_10, https://doi.org/10.1007/978-3-540-78440-1_10
22. Poelstra, A.: Scriptless scripts. Presentation Slides (3 2017), https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2017-03-mit-bitcoin-expo/slides.pdf, accessed on May 2024
23. Poelstra, A.: Mimblewimble and scriptless scripts. Presentation Slides (1 2018), https://download.wpsoftware.net/bitcoin/wizardry/mw-slides/2018-01-10-rwc/slides.pdf, accessed on May 2024
24. Schnorr, C.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO '89, Santa Barbara, California, USA, August 20-24, 1989, Proceedings. Lecture Notes in Computer Science, vol. 435, pp. 239–252. Springer (1989). https://doi.org/10.1007/0-387-34805-0_22, https://doi.org/10.1007/0-387-34805-0_22
25. Schnorr, C.: Efficient identification and signatures for smart cards (abstract). In: Quisquater, J., Vandewalle, J. (eds.) EUROCRYPT '89, Houthalen, Belgium, April 10-13, 1989, Proceedings. Lecture Notes in Computer Science, vol. 434, pp. 688–689. Springer (1989). https://doi.org/10.1007/3-540-46885-4_68, https://doi.org/10.1007/3-540-46885-4_68
26. Tairi, E., Moreno-Sanchez, P., Maffei, M.: Post-quantum adaptor signature for privacy-preserving off-chain payments. In: Borisov, N., Díaz, C. (eds.) FC 2021, Virtual Event, March 1-5, 2021, Revised Selected Papers, Part II. Lecture Notes in Computer Science, vol. 12675, pp. 131–150. Springer (2021). https://doi.org/10.1007/978-3-662-64331-0_7, https://doi.org/10.1007/978-3-662-64331-0_7
27. Tairi, E., Moreno-Sanchez, P., Schneidewind, C.: Ledgerlocks: A security framework for blockchain protocols based on adaptor signatures. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) CCS 2023, Copenhagen, Denmark, November 26-30, 2023. pp. 859–873. ACM (2023). https://doi.org/10.1145/3576915.3623149, https://doi.org/10.1145/3576915.3623149

28. Tu, B., Zhang, M., Yu, C.: Efficient ecdsa-based adaptor signature for batched atomic swaps. In: Susilo, W., Chen, X., Guo, F., Zhang, Y., Intan, R. (eds.) ISC 2022, Bali, Indonesia, December 18-22, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13640, pp. 175–193. Springer (2022). https://doi.org/10.1007/978-3-031-22390-7_12, https://doi.org/10.1007/978-3-031-22390-7_12

Public-Key Cryptography

QuietOT: Lightweight Oblivious Transfer with a Public-Key Setup

Geoffroy Couteau[1,2](✉), Lalita Devadas[3], Srinivas Devadas[3], Alexander Koch[1,2], and Sacha Servan-Schreiber[3]

[1] CNRS, Paris, France
[2] IRIF, Université Paris Cité, Paris, France
couteau@irif.fr
[3] MIT, Cambridge, MA, USA

Abstract. Oblivious Transfer (OT) is at the heart of secure computation and is a foundation for many applications in cryptography. Over two decades of work have led to extremely efficient protocols for evaluating OT instances in the preprocessing model, through a paradigm called OT extension. A few OT instances generated in an offline phase can be used to perform many OTs in an online phase efficiently, i.e., with very low communication and computational overheads.

Specifically, traditional OT extension protocols use a small number of "base" OTs, generated using any black-box OT protocol, and convert them into many OT instances using only lightweight symmetric-key primitives. Recently, a new paradigm of OT with a *public-key setup* has emerged, which replaces the base OTs with a non-interactive setup: Using only the public key of the other party, two parties can efficiently compute a virtually unbounded number of OT instances on-the-fly.

In this paper, we put forth a novel framework for OT extension with a public-key setup (henceforth, "public-key OT") and concretely efficient instantiations. Implementations of our framework are 30–100× faster when compared to the previous state-of-the-art public-key OT protocols, and remain competitive even when compared to OT protocols that *do not* offer a public-key setup. Additionally, our instantiations result in the first public-key schemes with plausible post-quantum security.

In summary, this paper contributes:

- QuietOT: A framework for OT extension with a public-key setup that uses fast, symmetric-key primitives to generate OT instances following a one-time public-key setup, and offering additional features such as precomputability.
- A public-key setup for QuietOT from the RingLWE assumption, resulting in the first post-quantum construction of OT extension with a public-key setup.
- An optimized, open-source implementation of our construction that can generate up to 1M OT extensions per second on commodity hardware. In contrast, the state-of-the-art public-key OT protocol is limited to approximately 20K OTs per second.
- The first formal treatment of the security of OT with a public-key setup in a multi-party setting, which addresses several subtleties that were overlooked in prior work.

© International Association for Cryptologic Research 2025
K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15485, pp. 197–231, 2025.
https://doi.org/10.1007/978-981-96-0888-1_7

1 Introduction

In its simplest form, Oblivious Transfer (OT) allows a party (called the *receiver*) to privately retrieve one out of two messages from another party (called the *sender*). The receiver has a choice bit b and the sender has a pair of messages (m_0, m_1). Using OT, the receiver learns m_b but learns nothing about m_{1-b}. Moreover, the sender is guaranteed to learn nothing about b. OT is a foundational building block for secure multiparty computation [50], and its applications typically require a large number of oblivious transfers (in the millions or billions). Unfortunately, all existing protocols for OT require public-key cryptography, making them concretely inefficient in many applications. Since it is known that OT cannot be constructed in a black-box manner using only symmetric-key primitives [45], this inefficiency is somewhat inherent to the OT problem. Fortunately, however, since the seminal work of Beaver [11] and the efficient construction of Ishai, Kilian, Nissim, and Petrank [47] (henceforth, IKNP), lightweight OT can be realized by performing a small number of expensive "base" OTs that are then extended (using only lightweight, symmetric-key cryptography) or "reused" to perform any number of regular OTs. Despite its concrete computational efficiency, the original paradigm of IKNP induces a large communication overhead (λ bits of communication per extended OT). To address this overhead, new paradigms have recently emerged that enable extending OTs using much less communication. Protocols like SoftSpoken OT [59] directly improve the communication efficiency of IKNP by a small factor k (e.g., $k = 5$) at the cost of some increased computation. Silent OT extension protocols [17,18,20,33,55,57,60,65] achieve optimal communication (3 bits of communication per OT), but come with a concrete computational overhead that is noticeably larger than SoftSpoken OT (e.g., RRT, the state-of-the-art silent OT [57], being about $8\times$ slower than SoftSpoken OT on machines with AVX instructions).

Our goal: "Diffie-Hellman" for Secure Computation. The state-of-the-art techniques for efficiently evaluating a large number of OT instances all require the sender and the receiver to initially interact in a distributed setup phase. Contrast this with the simpler task of establishing a secure *communication* channel on the Internet. Thanks to the breakthrough key-agreement protocol of Diffie and Hellman [37] in 1976, any pair of parties can locally derive a shared symmetric encryption key directly from the *public key* of the other party. This approach to securing communication has proven to be highly effective, and is now widely deployed [62]. Concretely, this means that over a large network of N parties, all pairs of parties can securely communicate following a one-time public-key setup with $O(N)$ communication, where all parties broadcast their public keys.

The goal of *oblivious transfer with a public-key setup*, first explicitly put forth by Orlandi, Scholl, and Yakoubov [55], is to achieve a similar feature for the task of secure *computation* on the Internet. Concretely, over a large network of N parties, if all pairs of parties want to be able to jointly run secure *computation* protocols (which typically requires evaluating many OTs), they must all run the distributed setup pair-wise, resulting in $O(N^2)$ communication and simultaneous interactions. However, with a *public-key setup* (or non-interactive "public-key OT" for short), each pair of parties can instead efficiently generate an arbitrary number of pseudorandom OT instances, given only the public key of the other party! These pseudorandom OT instances can then be derandomized in one round to perform a regular OT [12,25,55]. Unfortunately, despite being very desirable, this feature is not achievable with any of the state-of-the-art OT extension protocols, even in the semi-honest model.

Very recently, however, the work of Bui, Couteau, Meyer, Passelègue, and Riahinia [25] (henceforth, BCMPR) achieved the first practically efficient candidate construction of public-key OT by building a new Pseudorandom Correlation Function (PCF) for the OT correlation, and showing that it admits a public-key setup. Concretely, with a public-key PCF, two parties can, given only each other's public key, locally generate an arbitrary amount of pseudorandom OTs. In turn, these pseudorandom OTs can be used to perform a regular bit-OT in one round of interaction and three bits of communication). While this represents significant progress, their result falls short of providing a fully satisfactory solution to the problem of efficient public-key OT. For one, their protocol is *not* an OT extension, given that it requires (local) public-key operations for *every* OT that it generates. Consequently, it is considerably less efficient than state-of-the-art OT extension protocols. Concretely, BCMPR can generate up to 21K OTs per second, whereas state-of-the-art OT extension protocols can generate several million OTs per second [59]. Additionally, BCMPR is built around group-based primitives, making it not post-quantum secure, and relies on a new assumption they call "Sparse-power DDH" (or SPDDH for short) which is only proven secure in the generic group model.

1.1 Our Contributions

In this paper, we make several contributions, which we highlight here and describe in depth in our technical overview of Sect. 2. The primary contribution of this paper is QuietOT: a novel framework for fast OT extension with a public-key setup. With QuietOT, given only each other's public key, two parties can generate an arbitrary amount of pseudorandom "*ListOTs*," a variant of OT which we introduce, which can be converted into pseudorandom (resp. regular) OTs in one round and a small overhead in communication, e.g., using 4 bits/OT (resp. 7 bits/OT) in one of our instantiations. The only difference between our approach via ListOT and a standard PCF for OT is that the derandomization step incurs slightly more communication (e.g., 7 bits instead of 3 bits). Unlike all prior public-key OT protocols, QuietOT does *not* require public-key operations when generating OTs, making the concrete performance *one to six orders* of magnitude faster compared to the state-of-the-art OT protocols that offer a public-key setup. Moreover, the public-key setup of QuietOT can be replaced by a simple two-round setup using any black-box base OTs, yielding new constructions of two-round OT extension. Additionally, we show that the base OTs can be replaced with a public-key setup under the standard RingLWE assumption (with a superpolynomial modulus-to-noise ratio), allowing parties to non-interactively derive a shared key from which they can generate OT extensions.

We note that state-of-the-art OT extension protocols, such as SoftSpoken [59], remain significantly faster than QuietOT (e.g., about 7× faster in the regime where SoftSpoken communicates 16 bits/OT). The core advantage of QuietOT over these alternatives lies in its public-key setup: concretely, using QuietOT, two parties can execute the vast majority of the computation *before they even interact*, given only each other's public key. The interactive phase that follows involves solely cheap, non-cryptographic operations (a few XORs per OT). In contrast, using SoftSpoken or any state-of-the-art OT extension, the parties must *first* interact (to generate base OTs) *before* they can run the bulk of the computation and interact again to complete the protocol; this can cause significant delays during which both parties have to stay online. We believe that this precomputation feature of QuietOT is highly desirable in the setting of on-demand pairwise secure computation over a large network. As a bonus, QuietOT communicates less than SoftSpoken, and requires less rounds of interaction.

Under the hood, our framework combines any "Inner-Product Membership" weak PRF (IPM-wPRF) [25] with a Shiftable Constrained Pseudorandom Function (ShCPRF), a new primitive that we introduce in Sect. 5. Prior work [25,55] requires using public-key operations for each OT, translating to expensive group operations under either the Quadratic Residuosity (QR) or DDH assumption (the construction of Orlandi, Scholl, and Yakoubov [55], henceforth OSY, is mostly of theoretical interest due to the large number of group exponentiations required). In contrast, we show that a ShCPRF can be constructed unconditionally in the random oracle model by exploiting a recent CPRF construction [61]. In addition, because IPM-wPRF are lightweight symmetric-key primitives (i.e., they do not require public-key operations to evaluate), our overall OT extension protocol is very efficient. We provide a comparison to related work in Table 1.

Table 1. An overview of OT extension protocols and their maximum throughput observed across different hardware and parameter settings (full evaluation results provided in Sect. 7). PKS and PQ indicate whether the construction has a *public-key setup* and is plausibly *post-quantum* secure, respectively. For efficiency, nearly all OT extension protocols are instantiated in the Random Oracle Model (ROM). Note that in these constructions, the random oracle assumption can be generically replaced with a suitable correlation-robust hash function. †PKS not implemented.

	OT/s Max. Throughput	**Bits/OT** Communication	**PKS†**	**PQ**	**Assumptions**
IKNP	34,000,000	128	✗	✓	ROM
SoftSpoken ($k = 2$)	53,000,000	64	✗	✓	ROM
SoftSpoken ($k = 8$)	9,500,000	16	✗	✓	ROM
RRT	6,900,000	3	✗	✓	EC-LPN+ROM
OSY	1	3	✓	✗	QR+ROM
BCMPR	21,000	3	✓	✗	IPM-wPRF+SPDDH+ROM
QuietOT	1,200,000	7	✓	✓	IPM-wPRF+ROM

Additional Contributions. In addition to the main contribution of the QuietOT framework, this paper contributes:

- The first formal treatment of public-key OT when used in a secure multi-party computation over a large network. Our definitions and analysis address several subtle issues with using any public-key OT constructions (including BCMPR and OSY) in a multi-party setting where security must be guaranteed with respect to an adversary corrupting a subset of parties.
- A definition and construction of shiftable CPRFs, a twist on CPRFs where the master key holder can efficiently "shift" the constraint when evaluating the CPRF. This construction plays a crucial role in our framework and may be of independent interest.
- An open-source, optimized implementation of the BCMPR protocol (Bui et al. [25] do not provide an implementation of their public-key PCF), which we evaluate and compare to QuietOT in Sect. 7.

2 Technical Overview

In this section, we provide a detailed overview of our results. In Sect. 2.1, we start by covering the state-of-the-art BCMPR framework for public-key OT. Then, in Sect. 2.2, we cover the main ideas behind our QuietOT framework and the different realizations of it. In Sect. 2.3, we show how QuietOT implies two-round OT extension and full pre-computability for either the sender or the receiver. In Sect. 2.4, we explain our approach to non-interactive public-key setup under the RingLWE assumption. In Sect. 2.5, we overview our definitions of public-key setup with *multi-instance* security, which becomes a crucial building block for applying public-key OT to a multi-party computation setting.

2.1 Background on the BCMPR Framework

The BCMPR framework constructs a pseudorandom correlation function (PCF) for OT correlations using an IPM-wPRF. In addition to the IPM-wPRF requirement, they also require the Sparse-Power DDH (SPDDH) assumption and instantiate their public-key setup from the DCR assumption. In their PCF construction, the sender and receiver can compute OT correlations on-demand: The sender outputs two pseudorandom bits (s_0, s_1) while the receiver outputs (b, s_b), where $b \in \{0, 1\}$ is a pseudorandom choice bit. This correlation can then be converted into a chosen-bit OT with 3 bits of communication in one round of interaction using the transformation of Beaver [12].

At the heart of the BCMPR framework is a Constrained PRF (CPRF) $F = (F.\mathsf{KeyGen}, F.\mathsf{Eval}, F.\mathsf{Constrain}, F.\mathsf{CEval})$ with the constraint predicate set to a weak PRF[1] $f_{\mathbf{z}}$ that outputs a pseudorandom bit. At a high level, a CPRF has two keys: a master key and a constrained key. The constrained key only allows evaluating the PRF when the constraint predicate is satisfied. Hence, the weak

[1] A weak PRF is only pseudorandom on *uniformly random* inputs.

PRF f indicates if the input to F is constrained or not, making roughly half the inputs to F constrained. It was known (somewhat folklore) that any CPRF with a weak PRF as a constraint predicate can be used to construct a PCF for OT correlations [10]. However, prior to BCMPR, all existing CPRF constructions were either not sufficiently expressive to evaluate a weak PRF as a predicate or not concretely efficient enough to result in practical solutions [6,23,24,26,27,32,56]. Therefore, at the core of BCMPR is a construction of a CPRF *just* powerful enough to evaluate a suitable weak PRF candidate as the constraint predicate while remaining concretely efficient in practice. They realize such a CPRF by adapting the classical Naor-Reingold PRF [54]. In a nutshell, the BCMPR framework for OT correlations combines:

– A CPRF supporting a special class of *Inner-Product Membership* (IPM) constraints (the constrained key can evaluate the PRF on x if and only if $\langle \mathbf{z}, x\rangle \in S$, for some constraint vector $\mathbf{z} \in \mathcal{R}^n$ defined over a finite ring $\mathcal{R}$ and fixed set S partitioning the ring elements)[2], and
– Any weak PRF $f_{\mathbf{z}} : \{0,1\}^n \to \{0,1\}$ having an evaluation function that can be described as an inner-product membership predicate (which they call an IPM-wPRF). That is, $f_{\mathbf{z}}(x) = 0$ iff $\langle \mathbf{z}, x\rangle \in S_0$ and $f_{\mathbf{z}}(x) = 1$ iff $\langle \mathbf{z}, x\rangle \in S_1$, for a partitioning $S_0 \cup S_1$ of the finite ring $\mathcal{R}$ over which the inner product is defined, and a vector $\mathbf{z} \in \mathcal{R}^n$.

Building on the Naor-Reingold PRF, BCMPR constructs a CPRF supporting IPM predicates in the Random Oracle Model (ROM) [13]. In particular, using any IPM-wPRF (which can be realized from a handful of assumptions), coupled with their CPRF construction supporting IPM predicates, allows them to instantiate the following generic template for building a PCF for OT correlations, which will serve as inspiration for our framework as well.

A General PCF Template from CPRFs for IPM Predicates. The template of BCMPR uses a CPRF F with IPM constraints that evaluates an IPM-wPRF $f_{\mathbf{z}}$ as the predicate, for a vector $\mathbf{z} \in \mathcal{R}^n$ that we will call the wPRF key. The sender gets two master keys $(\mathsf{msk}_0, \mathsf{msk}_1)$ for F, while the receiver obtains two constrained keys $(\mathsf{csk}_0, \mathsf{csk}_1)$. The master keys can be used to evaluate the PRF on the entire domain. In contrast, the constrained key can only be used to evaluate the PRF when the constraint predicate is satisfied. The idea is to have the constrained key csk_0 have $f_{\mathbf{z}}$ as the predicate, and csk_1 have the opposite predicate $1 - f_{\mathbf{z}}$. Notice that given the two constrained keys, the receiver can only evaluate the CPRF using *one* of the two keys for an input x (depending on the value of $f_{\mathbf{z}}(x)$, which is pseudorandom). Moreover, given the IPM-wPRF key $\mathbf{z}$, the receiver can determine which of the two evaluations is constrained for an input x by evaluating the "predicate" $f_{\mathbf{z}}(x)$. The receiver can then compute and output the correlation (b, s_b), consisting of the pseudorandom bit $b = f_{\mathbf{z}}(x)$ and the string $s_b = F.\mathsf{CEval}(\mathsf{csk}_b, x)$. The sender, in contrast, only obtains the master keys $(\mathsf{msk}_0, \mathsf{msk}_1)$, which are independent of the IPM-wPRF key $\mathbf{z}$. As such, the sender can only compute the strings $s_0 = F.\mathsf{Eval}(\mathsf{msk}_0, x)$

[2] We slightly abuse notation by interpreting the bit string x as a *vector* of bits.

and $s_1 = F.\mathsf{Eval}(\mathsf{msk}_1, x)$, consisting of the sender's correlation (s_0, s_1), without learning the pseudorandom bit b computed by the receiver.

Limitations of the General Template. The core difficulty associated with the above template (and the BCMPR framework by extension) is finding a CPRF with a predicate class that is sufficiently powerful to evaluate $f_{\mathbf{z}}$. The most efficient construction to date is the constrained Naor-Reingold PRF construction of BCMPR, which (1) requires a new cryptographic assumption, (2) is not post-quantum secure and, (3) necessitates concretely expensive group operations to evaluate, placing an upper limit on practical efficiency of BCMPR (e.g., 21K correlations per second in our optimized implementation). In contrast, OT extension protocols like SoftSpoken OT [59] (which generalizes IKNP) use only lightweight symmetric-key primitives, are post-quantum secure, and are incredibly fast (e.g., achieving several million correlations per second), but do not offer public-key setups. Unfortunately, improving the efficiency of the BCMPR framework hinges on developing more efficient CPRF constructions for IPM predicates, which appears to be the weakest class of predicates sufficiently powerful to evaluate any wPRF. Note that a pseudorandom function *cannot* have a linear evaluation, and therefore inner-product equality predicates are inherently insufficient.

2.2 Our Approach

Intuition. The starting point of our approach is the template construction of BCMPR. As with BCMPR, in our framework, the receiver holds the key $\mathbf{z} \in \mathcal{R}^n$ of an IPM-wPRF and we let the (pseudorandom) selection bit of the receiver be defined as the output $f_{\mathbf{z}}(x)$ of the wPRF f on a random input x. Recall that $f_{\mathbf{z}}(x) = b$ iff $\langle \mathbf{z}, x \rangle \in S_b$ (where S_0, S_1 are a public partitioning of the inner-product range, associated with the IPM-wPRF). The main limitation of BCMPR is their reliance on a CPRF for a class of constraints that contains $f_{\mathbf{z}}$. While they provide an optimized construction, it still requires public-key operations (group exponentiation) for *every* evaluation, and hence for every OT instance.

At this point, we diverge significantly from the BCMPR framework by replacing their CPRF with a far more efficient primitive. Our starting point is a recent CPRF construction of Servan-Schreiber [61], which uses only symmetric-key primitives. Concretely, evaluating the CPRF involves computing an inner product and hashing the result; furthermore, the CPRF was shown to be unconditionally secure in the ROM. However, the catch is that the CPRF of Servan-Schreiber only handles inner-product predicates: That is, given a constraint $\mathbf{z}$, the constrained evaluation with csk on x matches the evaluation with the master key if and only if $\langle \mathbf{z}, x \rangle = 0 \in \mathcal{R}$. Observe that using this much weaker CPRF, the receiver is now only able to evaluate F on all inputs where $\langle \mathbf{z}, x \rangle = 0$[3] (roughly $\frac{1}{|S_b|}$ of all inputs assuming $f(x) = b$, and where $|S_b| \approx |\mathcal{R}|/2$), which is too weak to instantiate the BCMPR template.

[3] We follow the convention of letting $P(x) = 0$ when the predicate P is satisfied.

Shiftable CPRFs to the Rescue. Our first key observation is that (a slight modification of) the CPRF framework of Servan-Schreiber enjoys an additional *shiftability* property. Concretely, the CPRF evaluation with the master key msk can take an additional *shift* α as input, and provides the following guarantee: The constrained evaluation $F.\mathsf{CEval}(\mathsf{csk}, x)$ is equal to $F.\mathsf{Eval}(\mathsf{msk}, x, \alpha)$ whenever $\langle \mathbf{z}, x\rangle - \alpha = 0$. That is, the constraint is *shifted by* α. Given such a shiftable CPRF for inner products, the sender can now compute $F.\mathsf{Eval}(\mathsf{msk}, x, \alpha)$ for *all possible shifts* $\alpha \in S_0 \cup S_1$. This yields two lists of values: $L_0 = (F.\mathsf{Eval}(\mathsf{msk}, x, \alpha))_{\alpha \in S_0}$ and $L_1 = (F.\mathsf{Eval}(\mathsf{msk}, x, \alpha))_{\alpha \in S_1}$. Our next core observation is that the value $F.\mathsf{CEval}(\mathsf{csk}, x)$ computed by the receiver belongs to exactly one of the two lists, and furthermore, *the index* b *of the list* L_b *it belongs to is simply* $f_{\mathbf{z}}(x)$. That is, the receiver knows a pseudorandom value v and pseudorandom "selection bit" $b = f_{\mathbf{z}}(x)$ such that $v \in L_b$. Additionally, by using the constraint $\mathbf{z}$, the receiver can determine the index i in L_b in which v is located (i.e., such that $v = L_b[i]$).

Oblivious Transfer from ListOT. So far, we have seen that given a shiftable CPRF for inner-product predicates, the sender and the receiver can generate many instances of the following "correlation:" The sender gets as output two (pseudorandom) lists (L_0, L_1), and the receiver obtains (v, b, i) where $v = L_b[i]$, and b is pseudorandom from the viewpoint of the sender (however, importantly, i is *not* pseudorandom, which prevents this from being a true OT correlation). We call "ListOT" this weaker variant of the OT correlation. The name is inspired from *list decoding* [39], where a decoding algorithm for a code is allowed to output a list of code words from which the word can be decoded.[4] Hence, for ListOT, the sender outputs two lists of messages, L_0, L_1, and the receiver outputs a bit b, value v, *and* an index key i, such that v is located at $L_b[i]$. (Later, for ease of notation, L_0 and L_1 will be treated as key-value stores/dictionaries.)

While the pseudorandom ListOT instances are *not* correlations in the strict technical sense (because the distribution of i depends on the secret wPRF key), it is not too hard to see that it still suffices to instantiate a random OT using some additional communication. To see this, observe that given OT inputs (m_0, m_1), the sender simply sends $(L_0[j] \oplus m_0)_{j \in S_0}$ and $(L_1[j] \oplus m_1)_{j \in S_1}$ to the receiver. The receiver recovers m_b by unmasking $L_b[i] \oplus m_b$ using $v = L_b[i]$.[5]

We now explain how we construct efficient ShCPRFs by adapting the framework of Servan-Schreibe [61] building CPRFs for inner-product predicates from RKA-secure PRFs [14] in the standard model (or in the random oracle model).

Constructing ShCPRFs. We make the observation that in all existing CPRF constructions for inner-product predicates [25,35,61], the master key holder can efficiently compute the set of all possible pseudorandom values evaluated under

[4] We note that ListOT is not related to "list two-party computation" [28], which defines list OT as a security definition for the standard oblivious transfer functionality.

[5] When generating (pseudo)random OTs, this simple approach can be further improved by letting m_0 and m_1 be the first element of L_0 and L_1 respectively, which allows communicating two elements less, for a total of $|S_0| + |S_1| - 2$ bits of communication. Concretely, with our most communication-efficient instance, this translates to only 4 bits of communication per random OT.

the constrained key csk. We will focus on the CPRF construction of Servan-Schreiber, instantiated unconditionally using a hash function H modeled as a random oracle. In this construction, the master key msk consists of a random vector $\mathbf{z_0}$ of length n, with elements from some sufficiently large field $\mathbb{F}$.[6] For a constraint vector $\mathbf{z} \in \mathbb{F}^n$, the constrained key is defined as $\mathbf{z_1} = \mathbf{z_0} - \Delta \cdot \mathbf{z}$, where $\Delta \in \mathbb{F} \setminus \{0\}$ is random. Simplifying slightly,[7] the evaluation and the constrained evaluation algorithms are defined as $H(\langle \mathbf{z_0}, x\rangle, x)$ and $H(\langle \mathbf{z_1}, x\rangle, x)$, respectively. Note that when $\langle \mathbf{z}, x\rangle = 0$, it holds that $H(\langle \mathbf{z_0}, x\rangle, x)$ is equal to $H(\langle \mathbf{z_1}, x\rangle, x)$, which guarantees the master key and constrained key evaluations agree. In contrast, when $\langle \mathbf{z}, x\rangle \neq 0$, then $H(\langle \mathbf{z_1}, x\rangle, x)$ is equal to $H(\langle \mathbf{z_0}, x\rangle - \Delta \langle \mathbf{z}, x\rangle, x)$, which is independent of $H(\langle \mathbf{z_0}, x\rangle, x)$ due to Δ. In particular, we observe that when $\langle \mathbf{z}, x\rangle \neq 0$, using $\mathbf{z_0}$ and Δ allows the master key holder to evaluate *all possible* constrained evaluations by computing $H(\langle \mathbf{z_1}, x\rangle + \Delta\alpha, x)$, for all possible inner products $\alpha \in \{\langle \mathbf{z}, x\rangle \mid x \in \{0,1\}^n\}$ associated with the constraint class given by $\mathbf{z}$. We point to Sect. 4 for more details on this ShCPRF construction. Abstractly, we define the master key evaluation algorithm $F.\mathsf{Eval}(\mathsf{msk}, x, \alpha)$ to take a shift α as an additional input while leaving the remaining CPRF algorithms unchanged.

Putting Things Together: A "PCF" for ListOT. Using the ShCPRF construction sketched above, coupled with an IPM-wPRF $f_{\mathbf{z}}$ with partitioning $S_0 \cup S_1$, the sender with the master secret key msk computes the two lists, L_0 and L_1, corresponding to $b = 0$ and $b = 1$, respectively, as $L_0 = (F.\mathsf{Eval}(\mathsf{msk}, x, \alpha))_{\alpha \in S_0}$, $L_1 = (F.\mathsf{Eval}(\mathsf{msk}, x, \beta))_{\beta \in S_1}$, using a random x. Importantly, note that given the constrained key csk for a constraint vector $\mathbf{z}$, the receiver obtains *one* value in L_b, where $b = f_{\mathbf{z}}(x)$. All other values, in both lists, remain pseudorandom from the viewpoint of the receiver. At this stage, our framework can be instantiated using any choice of ShCPRF and any choice of IPM-wPRF. We choose to instantiate the ShCPRF in the random oracle model, as it offers the most concretely-efficient solution. The IPM-wPRF can be realized from several assumptions, as detailed in BCMPR. Indeed, many wPRFs fit the IPM-wPRF framework, including the Learning-with-Rounding (LWR)-based wPRF [8], the Goldreich-Applebaum-Raykov (GAR) [4,41], and several other low-complexity wPRF candidates, including the Boneh, Ishai, Passelègue, Sahai, and Wu (BIPSW) [15], and LPN-based candidates [19]. (Bui et al. [25] provide an overview of these different candidates and others). The BIPSW wPRF candidate is especially well-suited to this framework given that the evaluation (defined in Equation (1)) is essentially just a rounded inner product computed in $\mathbb{Z}_6$:

$$f_{\mathbf{z}}(x) = \lfloor \langle \mathbf{z}, x\rangle \mod 6\rceil_2 . \tag{1}$$

Note that when $f_{\mathbf{z}}(x) = 0$, then it holds that $\langle \mathbf{z}, x\rangle \pmod 6 \in \{0,1,2\}$ and when $f_{\mathbf{z}}(x) = 1$ it holds that $\langle \mathbf{z}, x\rangle \pmod 6 \in \{3,4,5\}$. By instantiating the ShCPRF to compute predicates over an extension of $\mathbb{Z}_6$, we can achieve incredibly efficient evaluations using the BIPSW IPM-wPRF (see Sect. 7 for our evaluation).

[6] Our actual ShCPRF construction is defined using a ring extension for efficiency.
[7] The full construction has an extra additive term to handle the all-zero input $x = 0^n$.

2.3 Two-Round OT Extension

Using our framework, we obtain a *two-round* OT extension protocol. Observe that the sender can independently generate the ShCPRF master secret key, consisting of $\mathbf{z_0}$ and Δ, while the receiver can independently generate the IPM-wPRF key $\mathbf{z}$. For the case where $\mathbf{z} \in \{0,1\}^n$, we can use any two-round string OT protocol repeated in parallel n times as follows. For $i \in [n]$, the sender sets $m_{i,0} = \mathbf{z_0}_i$ and $m_{i,1} = \mathbf{z_0}_i - \Delta$. The receiver uses $\mathbf{z}_i \in \{0,1\}$ as its choice bit to retrieve $m_{i,\mathbf{z}_i} = \mathbf{z_0}_i - \Delta \mathbf{z}_i$, and in this way can recover $\mathsf{csk} := \mathbf{z_0} - \Delta \mathbf{z}$ using n parallel calls to the two-round OT functionality (indeed, because Δ is the same across messages, any *correlated* OT protocol [5] is sufficient). In the general case, when $\mathbf{z} \in \mathcal{R}^n$, we can use any two round "reverse" Vector Oblivious Linear Evaluation (VOLE) protocol [2,18], which directly generalizes correlated OT to work over a ring $\mathcal{R}$. In reverse VOLE, the sender inputs $(\mathbf{b}, x) \in \mathcal{R}^n \times \mathcal{R}$ and the receiver inputs $\mathbf{a} \in \mathcal{R}^n$. The sender receives no output while the receiver obtains $\mathbf{a}x + \mathbf{b}$. By letting the sender input $(\mathbf{z_0}, \Delta)$ and the receiver input $-\mathbf{z}$, we immediately have that the receiver obtains $\mathbf{z_1} = \mathbf{z_0} - \Delta \mathbf{z}$. See the full version [29] for more details.

Two-round OT extension is known to be impossible under black-box symmetric-key primitives [40] making our use of an IPM-wPRF a rather weak assumption to circumvent the impossibility result of Garg et al. [40] (in fact, an IPM-PR**G** suffices). In contrast, protocols like IKNP and SoftSpoken inherently require three rounds of interaction due to their unconditional instantiations in the random oracle model, and all previous two-round OT extensions (with the exception of Beaver [11], which is not black-box and not concretely efficient) required variants of the LPN assumptions [17,18,57,65].

Precomputability. A nice feature of our two-round setup is the ability for one party to *precompute* all correlations *before even knowing the identity of the other party.* To see this, note that the receiver can precompute all choice bits just using the IPM-wPRF key $\mathbf{z}$ without needing to know the constrained key. Additionally, the receiver can sample a *uniformly random* constrained key $\mathbf{z_1}$ for the ShCPRF and use it to generate ahead-of-time all its ListOT triples (b, i, v). Later, once the identity of the sender is known, the sender can engage with the receiver in a two-round OT protocol to compute the master key $\mathbf{z_0} = \mathbf{z_1} + \Delta \mathbf{z}$ *from* the "constrained key" $\mathbf{z_1}$. Similarly, the sender can alternatively generate all its ListOT instances (L_0, L_1) without needing to know the identity of the receiver by locally sampling Δ and $\mathbf{z_0}$. We provide details on precomputability and more motivation for the notion in the full version [29].

2.4 Public-Key Setup from Ring-LWE

We present a non-interactive distributed setup protocol from RingLWE. To the best of our knowledge, this forms the first distributed setup protocol for PCF for ListOT based on a plausibly post-quantum assumption. The goal of this protocol is for the sender with input Δ and receiver with input $\mathbf{z}$ to distributively derive keys $\mathbf{z_0}$ (part of the master secret key) and $\mathbf{z_1}$ (the constrained key), which can be viewed as additive shares of $\Delta \cdot \mathbf{z}$ in a ring $\mathcal{R}$.

Parameters. In order to rely on the security of RingLWE, the receiver will "encode" the bits of $\mathbf{z} \in \mathcal{R}^n$ into the coefficients of an element z of a suitable polynomial ring $\mathcal{P}$. The protocol is executed over $\mathcal{P}$ and then, at the end of the protocol, the sender and receiver each "decode" their result back into the ring $\mathcal{R}$ to obtain vectors $\mathbf{z_0}$ and $\mathbf{z_1}$, by parsing each polynomial as a vector of n coefficients (and disregarding any extra coefficients).

Assume that $\mathcal{R} = \mathbb{Z}_t$ is an integer ring, and let $q := n \cdot t \cdot B \cdot 2^{\omega(\log \lambda)}$ (B is some bound on the noise that we compute later). We define $\mathcal{P} := \mathbb{Z}_q[X]/(X^\eta + 1)$, where η is a power of 2 that is larger than n. Let $\chi = \chi(\mathcal{P})$ be a suitable noise distribution over $\mathcal{P}$, such that for $e_0, e_1 \xleftarrow{\text{R}} \chi$, it holds that $\|e_0 e_1\|_\infty \leq B/3$, with overwhelming probability.

The protocol proceeds in two phases as follows. During the public-key generation phase, the sender and receiver each broadcast a public key, which is used by the other party in the ShCPRF evaluation key derivation phase.

Step 1: Generating Public Keys. Fix random $a_0, a_1 \in \mathcal{P}$ as part of the public parameters. To generate public keys, the sender and receiver proceed as follows. These public keys can then be posted to a bulletin board or broadcasted.

Sender
```
1: Sample secret $s_0 \xleftarrow{\text{R}} \chi$.
2: Sample error $e_0 \xleftarrow{\text{R}} \chi$.
3: Set $\mathsf{pk}_S = \Delta \cdot a_0 + s_0 a_1 + e_0$.
```

Receiver
```
1: Encode $\frac{q}{t} \cdot \mathbf{z}$ as $z \in \mathcal{P}$.
2: Sample secret $s_1 \xleftarrow{\text{R}} \chi$.
3: Sample errors $e_1, e_1' \xleftarrow{\text{R}} \chi$.
4: Set $\mathsf{pk}_R = (z + s_1 a_0 + e_1, s_1 a_1 + e_1')$.
```

Step 2: Deriving ShCPRF keys. To derive a master key msk and constrained key csk, respectively, the sender and receiver use the other party's public key to proceed as follows. (Here and throughout, we overload rounding $\lceil \cdot \rfloor_t$ notation to include "rounding" a polynomial coefficient-by-coefficient.)

Sender
```
1: Compute $z_0 := \lceil \langle \mathsf{pk}_R, (\Delta, s_0) \rangle \rfloor_t$.
2: Decode $z_0 \in \mathcal{P}$ as $\mathbf{z_0} \in \mathcal{R}^n$.
3: Set $\mathsf{msk} := (\mathbf{z_0}, \Delta)$.
```

Receiver
```
1: Compute $z_1 := \lceil \mathsf{pk}_S \cdot s_1 \rfloor_t$.
2: Decode $z_1 \in \mathcal{P}$ as $\mathbf{z_1} \in \mathcal{R}^n$.
3: Set $\mathsf{csk} := \mathbf{z_1}$.
```

Correctness.. The inner products computed in the key derivation phase are, in fact, noisy additive shares of $\Delta \cdot z \in \mathcal{P}$, since we have that

$$\begin{aligned}
&\langle \mathsf{pk}_R, (\Delta, s_0) \rangle - (\mathsf{pk}_S \cdot s_1) \\
&= \Delta \cdot z + \Delta \cdot a_0 s_1 + \Delta \cdot e_1 + s_0 a_1 s_1 + s_0 e_1' - \Delta \cdot a_0 s_1 - s_0 a_1 s_1 - e_0 s_1 \\
&= \Delta \cdot z + \underbrace{\Delta \cdot e_1 + s_0 e_1' - e_0 s_1}_{\text{noise}} \approx \Delta \cdot z.
\end{aligned}$$

Note that $\Delta \in \mathbb{Z}_t$ has low norm, so we can bound the magnitude of the noise term $\Delta \cdot e_1 + s_0 e_1' - e_0 s_1$ by B. Hence, by a standard rounding lemma [22,38], $z_0 - z_1 = \lceil \langle \mathsf{pk}_R, (\Delta, s_0) \rangle \rfloor_t - \lceil \mathsf{pk}_S \cdot s_1 \rfloor_t = \Delta \cdot z \bmod t$. After parsing as vectors over $\mathcal{R}^n$, we have $\mathbf{z_0} - \mathbf{z_1} = \Delta \cdot \mathbf{z}$.

Security.. Pseudorandomness of the public keys follows from the RingLWE assumption with short secrets (i.e., *normal form* RingLWE).[8] In the sender public key, the RingLWE sample $s_0 a_1 + e_0$ masks $\Delta \cdot a_0$ and thus the secret key Δ. Similarly, in the receiver public key, the RingLWE sample $s_1 a_0 + e_1$ masks z and thus the secret key $\mathbf{z}$. For a complete description of our protocol, its parameters, and proof of security, we refer to the full version [29].

2.5 Multi-instance Security

An immediate application of QuietOT (and public-key OT schemes in general [25,55]) is for efficient large-scale MPC. At a high level, with QuietOT, parties can, using just the public keys of all other parties, create pair-wise OT channels for the purpose of running a secure computation (e.g., as in the GMW protocol [42]). This application was also described in prior public-key OT constructions [25,55] but was never formalized. We make the rather subtle observation that existing definitions [25,55] for public-key OT only require *one-time* security—i.e., privacy for the sender or receiver is not considered when the same public keys are reused with different parties.

To address this gap and properly define public-key OT, we formalize the notion of "multi-instance security" in the full version of this work [29]. In a nutshell, our definition captures a setting where parties (re)use a *long-term* secret (that depends on the public-key) and an *ephemeral* secret that is generated for each new session. We then prove that our public-key setup satisfies multi-instance security.

3 Preliminaries

Notation. We let $\mathbb{N}$ denote the set of natural numbers and $\mathbb{F}_p$ denote a finite field of order p. We denote a vector $\mathbf{v} = (v_1, \ldots, v_n)$ using bold letters. We denote by $\mathsf{poly}(\cdot)$ any polynomial and by $\mathsf{negl}(\cdot)$ any negligible function. We let $x \overset{R}{\leftarrow} S$ denote a uniformly random sample drawn from S. We let $x \leftarrow \mathcal{A}$ denote assignment and $x := y$ denote initialization of x to the value of y. By an *efficient* algorithm $\mathcal{A}$ we mean that $\mathcal{A}$ is modeled by a (possibly non-uniform) Turing Machine that runs in probabilistic polynomial time. We write $D_0 \approx_c D_1$ to mean that two distributions D_0 and D_1 are *computationally* indistinguishable to all efficient distinguishers $\mathcal{D}$ and $D_0 \approx_s D_1$ to mean that D_0 and D_1 are *statistically* indistinguishable.

[8] Normal form RingLWE is a standard variant of RingLWE where the secret is sampled from the noise distribution instead of uniformly. It is known to be as hard as regular RingLWE and is often used for practical schemes [1,36,51,53].

3.1 Cryptographic Definitions

Here, we recall the cryptographic definitions that we will use throughout the paper. In Sect. 3.1.1, we define the notion of an IPM-wPRF. In Sect. 3.1.2, we cover the definition of Ring-LWE and the basics of modular rounding.

3.1.1 Inner-Product Membership wPRF

We define the notion of a weak PRF (wPRF)[9] that can be evaluated using the "inner-product membership" formalism introduced by Bui et al. [25].

Definition 1 (Inner-Product Membership wPRF (IPM-wPRF) [25]). *Let λ be the security parameter and $\mathcal{R} = \mathcal{R}$ be a finite ring. Let $S_0 = S_0^{(\lambda)}$ be a (polynomially-sized) subset of $\mathcal{R}_\lambda$, and set $S_1 := \mathcal{R} \setminus S_0$. Then, $f := f_\lambda \colon \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \{0,1\}$ is an inner-product membership weak PRF (IPM-wPRF) family with respect to the partitioning (S_0, S_1), if it satisfies the following three properties:*

(1) $\mathcal{K}_\lambda = \mathcal{X}_\lambda = \mathcal{R}_\lambda^n$ for some $n = n(\lambda)$,
(2) its evaluation can be expressed as an inner product membership, i.e., for each $\lambda \in \mathbb{N}$, $\mathbf{z} \in \mathcal{K}_\lambda$, $\mathbf{x} \in \mathcal{X}_\lambda$, we have that

$$f_{\mathbf{z}}(\mathbf{x}) = \begin{cases} 0, & \textit{if } \langle \mathbf{z}, \mathbf{x} \rangle \in S_0 \\ 1, & \textit{otherwise (i.e., } \langle \mathbf{z}, \mathbf{x} \rangle \in S_1), \end{cases}$$

where $\langle \cdot, \cdot \rangle$ is the standard (simple) inner product on $\mathcal{R}^n$, and
(3) it achieves the standard notion of a secure (weak) PRF [48].

3.1.2 Ring Learning with Errors and Rounding

We recall the standard Ring Learning-with-Errors (RingLWE) assumption of Lyubashevsky, Peikert, and Regev [52] and its normal form.

Definition 2 (The RingLWE assumption [52]). *Let $\eta = \eta(\lambda), q = q(\lambda) \in \mathbb{N}$. Define the polynomial ring $\mathcal{P} = \mathbb{Z}_q[X]/(X^\eta + 1)$ and let $\chi = \chi(\lambda)$ be an error distribution over $\mathcal{P}$. The $\mathsf{RingLWE}_{\eta,q,\chi}$ assumption states that for any $t = t(\lambda) \in \mathsf{poly}(\lambda)$, it holds that*

$$(\mathbf{a}, s \cdot \mathbf{a} + \mathbf{e}) \approx_c (\mathbf{a}, \mathbf{u}),$$

where $s \overset{R}{\leftarrow} \mathcal{P}, \mathbf{a} \overset{R}{\leftarrow} \mathcal{P}^t, \mathbf{e} \overset{R}{\leftarrow} \chi^t, \mathbf{u} \overset{R}{\leftarrow} \mathcal{P}^t$. The "normal form" $\mathsf{RingLWE}_{\eta,q,\chi}$ assumption states that this holds even when $s \overset{R}{\leftarrow} \chi$, and has the same hardness as the original formulation [51, Lemma 2.24].

Modular Rounding. We let $\lfloor x \rceil$ denote the rounding of a real number x to the nearest integer. For integers $q > p \geq 2$, we define the modular rounding function $\lfloor \cdot \rceil_p : \mathbb{Z}_q \to \mathbb{Z}_p$ as $\lfloor v \rceil_p = \lfloor (p/q) \cdot v \rceil$.

Rounding Lemma.. We recall the following "rounding lemma" [22,34,38]:

[9] A weak PRF is pseudorandom on *uniformly random* inputs.

Lemma 1 (Rounding of Noisy Secret Shares). *Let (t, q) be two integers such that t divides q. Fix any $z \in \mathbb{Z}_q$ and let (z_0, z_1) be any two random elements of $\mathbb{Z}_q$ subject to $z_0 + z_1 = (q/t) \cdot z + e \bmod q$, where e is such that $q/(t \cdot |e|) \geq \lambda^{\omega(1)}$. Then, with probability at least $1 - (|e| + 1) \cdot t/q \geq 1 - \lambda^{-\omega(1)}$, it holds that $\lfloor z_0 \rceil_t + \lfloor z_1 \rceil_t = z \bmod t$, and the probability is over the random choice of $(z_0, z_1) \in \mathbb{Z}_q \times \mathbb{Z}_q$.*

4 Shiftable CPRFs

In this section, we start by defining the notion of *Shiftable* CPRFs in Sect. 4.1. Then, in Sect. 4.2, we construct ShCPRFs for inner-product predicates by adapting the framework of Servan-Schreiber [61].

4.1 Defining Shiftable CPRFs

For simplicity, we restrict the definition to 1-key (rather than multi-key) ShCPRFs and selective security, which is the definition that is satisfied by our construction. The definition overlaps significantly with the definition of (non-shiftable) CPRFs [16,21,49] but using the classic PRF-style "real-or-random" security game, where all evaluations are either computed using the master key or using a truly random function.

Remark 1 (Relation to Shift-Hiding Shiftable Functions (SHSF)). SHSF are an extension to CPRFs introduced by Peikert and Shiehian [56]. In a SHSF, the constrained CPRF evaluation computes $F.\mathsf{Eval}(\mathsf{msk}, x) + f(x)$, for a hidden "shift" function f embedded into the constrained key. In contrast, our notion of Shiftable CPRFs only allows the master key holder to shift the *constraint* when evaluating the CPRF (using the master key) and does not affect the constrained key.

Definition 3 (Shiftable Constrained Pseudorandom Functions). *Let $\lambda \in \mathbb{N}$ be a security parameter. A Shiftable Constrained Pseudorandom Function (ShCPRF) with domain $\mathcal{X} = \mathcal{X}_\lambda$, range $\mathcal{Y}$, and a finite set of shifts $\mathcal{S}$ that supports constraints represented by the class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$, where $\mathcal{C}_\lambda \colon \mathcal{X} \times \mathcal{S} \to \{0, 1\}$, consists of the following four algorithms. We highlight the parts that are specific to shiftable CPRFs.*

- $\mathsf{KeyGen}(1^\lambda) \to \mathsf{msk}$. *Takes as input a security parameter λ. Outputs a master secret key* msk.
- $\mathsf{Eval}(\mathsf{msk}, x, \alpha) \to y$. *Takes as input the master secret key* msk*, an input $x \in \mathcal{X}$, and a shift $\alpha \in \mathcal{S}$. Outputs $y \in \mathcal{Y}$.*
- $\mathsf{Constrain}(\mathsf{msk}, C) \to \mathsf{csk}$. *Takes as input the master secret key* msk *and a constraint circuit $C \in \mathcal{C}$. Outputs a constrained key* csk.
- $\mathsf{CEval}(\mathsf{csk}, x) \to y$. *Takes as input the constrained key* csk *and an input $x \in \mathcal{X}$. Outputs $y \in \mathcal{Y}$.*

We let any public parameters PP *be an implicit input to all algorithms. A ShCPRF must satisfy the following correctness, security, and pseudorandomness properties. We let* $\widetilde{\mathcal{F}} = \widetilde{\mathcal{F}}_\lambda$ *denote the set of all functions from* $\mathcal{X} \times \mathcal{S}$ *to* $\mathcal{Y}$.

Correctness. *For all security parameters* λ, *all constraints* $C \in \mathcal{C}$, *and all inputs* $x \in \mathcal{X}$, *there exists an efficiently computable* $\alpha \in \mathcal{S}$ *such that* $C(x, \alpha) = 0$ *(authorized), and for all* $\alpha \in \mathcal{S}$ *where* $C(x, \alpha) = 0$ *it holds that:*

$$\Pr\left[\begin{array}{l}\mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \mathsf{csk} \leftarrow \mathsf{Constrain}(\mathsf{msk}, C)\end{array} : \mathsf{Eval}(\mathsf{msk}, x, \alpha) = \mathsf{CEval}(\mathsf{csk}, x)\right] = 1 - \mathsf{negl}(\lambda),$$

where the probability space is over the randomness used in KeyGen *and* $\mathsf{Constrain}$.

(1-Key, Selective) Security. *A ShCPRF is (1-key, selectively)-secure if for all efficient adversaries* $\mathcal{A}$, *the advantage of* $\mathcal{A}$ *in the following security experiment* $\mathsf{Exp}^{\mathsf{shcprf}}_{\mathcal{A},b}(\lambda)$ *is negligible in* λ. *Here,* b *denotes the challenge bit.*

1. ***Setup:*** *On input* 1^λ, *the challenger*
   - *runs* $\mathcal{A}(1^\lambda)$ *who outputs a constraint* $C \in \mathcal{C}$,
   - *computes* $\mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and* $\mathsf{csk} \leftarrow \mathsf{Constrain}(\mathsf{msk}, C)$,
   - *samples a uniformly random function* $R \xleftarrow{R} \widetilde{\mathcal{F}}_\lambda$, *and*
   - *sends* csk *to* $\mathcal{A}$.
2. ***Evaluation Queries:*** $\mathcal{A}$ *adaptively sends arbitrary inputs* $x \in \mathcal{X}$ *and shifts* $\alpha \in \mathcal{S}$ *to the challenger. For each pair* (x, α), *if* $C(x, \alpha) = 0$, *then the challenger returns* $\perp$. *Otherwise, the challenger proceeds as follows:*
   - *If* $b = 0$, *it computes* $y \leftarrow \mathsf{Eval}(\mathsf{msk}, x, \alpha)$ *and returns* y.
   - *If* $b = 1$, *it computes* $y \leftarrow R(x, \alpha)$ *and returns* y.
3. ***Guess:*** $\mathcal{A}$ *outputs its guess* b', *which is the output of the experiment.*

$\mathcal{A}$ *wins if* $b' = b$, *and its advantage* $\mathsf{Adv}^{\mathsf{shcprf}}_{\mathcal{A}}(\lambda)$ *is defined as*

$$\mathsf{Adv}^{\mathsf{shcprf}}_{\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{shcprf}}_{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{shcprf}}_{\mathcal{A},1}(\lambda) = 1]\right|,$$

where the probability is over the randomness of $\mathcal{A}$ *and* KeyGen.

Pseudorandomness. *A ShCPRF is said to be pseudorandom if for all efficient adversaries* $\mathcal{A}$, *it holds that*

$$\left|\Pr_{\mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda)}[\mathcal{A}^{\mathsf{Eval}(\mathsf{msk},\cdot,\cdot)}(1^\lambda) = 1] - \Pr_{R \xleftarrow{R} \widetilde{\mathcal{F}}_\lambda}[\mathcal{A}^{R(\cdot,\cdot)}(1^\lambda) = 1]\right| = 1 - \mathsf{negl}(\lambda).$$

Remark 2 (The Requirement for the Pseudorandomness Property). We note that while the pseudorandomness property is implicit in standard CPRF definition (which has a polynomial loss in security) [16], in the real-or-random style definition for CPRFs (as in Definition 3), this property is not immediately satisfied, since the challenger outputs $\perp$ when given an unconstrained query. Hence, we require the separate *pseudorandomness* property to capture the requirement that the function being constrained is indeed a standard PRF.

4.2 Constructing Shiftable CPRFs

In this section, we adapt the framework of Servan-Schreiber [61] constructing CPRFs for inner-product predicates from RKA-secure PRFs. We make the observation that the construction can be easily adapted to fit the shiftable CPRF definition (Definition 3). In the process, we additionally generalize the construction of Servan-Schreiber to work over a small ring as opposed to a large field, which makes it integrate better with an IPM-wPRF as the predicate.

The CPRF Framework of Servan-Schreiber in a Nutshell. The CPRF of [61] is parameterized by a security parameter λ, finite field $\mathbb{F}$ of order at least 2^λ, and a vector length parameter $n \geq 1$. The master secret key msk consists of a random vector $\mathbf{z_0} \in \mathbb{F}^n$. The constrained key csk for a constraint $\mathbf{z} \in \mathbb{F}^n$ is then defined as $\mathbf{z_1} := \mathbf{z_0} - \Delta\mathbf{z}$, with $\Delta \in \mathbb{F} \setminus \{0\}$ a random non-zero scalar. The main insight behind the framework of [61] is that for an input $\mathbf{x} \in \mathbb{F}^n$, when $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ (i.e., when the constraint is satisfied), then the inner product $\langle \mathbf{z_0}, \mathbf{x} \rangle$ is equal to $\langle \mathbf{z_1}, \mathbf{x} \rangle$. This fact can be used to derive *identical* PRF keys k and k' under both msk and csk:

$$k = \langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle - \cancel{\Delta \langle \mathbf{z}, \mathbf{x} \rangle} = \langle \mathbf{z_1}, \mathbf{x} \rangle = k'.$$

In contrast, when $\langle \mathbf{z}, \mathbf{x} \rangle \neq 0$, the $\Delta \langle \mathbf{z}, \mathbf{x} \rangle$-term makes $k \neq k'$. Moreover, because Δ is uniformly random over $\mathbb{F} \setminus \{0\}$ (where $\mathbb{F}$ has order at least 2^λ), $\mathbf{z_1}$ cannot be used to recover $\mathbf{z_0}$, even with knowledge of the constraint $\mathbf{z}$. The evaluation of the CPRF is then defined as $F_{k_0+k}(\mathbf{x})$ (resp. $F_{k_0+k'}(\mathbf{x})$ for the constrained evaluation), where k_0 is a "zero" PRF key used to handle the case where $\mathbf{x} = 0^n$. One caveat, however, is that the derived PRF keys are *highly correlated*, which necessitates choosing F to be a suitable RKA-secure PRF. The work of Servan-Schreiber shows that when the PRF F is RKA-secure for affine key-derivation functions (see the full version of our paper [29] for a definition), then the CPRF instantiated with the PRF F is secure. (We note that a random oracle $H\colon \mathbb{F} \times \mathbb{F}^n \to \{0,1\}^*$ is RKA-secure PRF for all non-trivial key-derivation functions.)

Adding Shiftability. We make the simple observation that if we make the master secret key msk also contain Δ, then the we can easily turn the above framework into a *shiftable* CPRF as follows. Specifically, when $\langle \mathbf{z}, \mathbf{x} \rangle \neq 0$, the constrained key computes $k' = \langle \mathbf{z_0}, \mathbf{x} \rangle - \Delta \langle \mathbf{z}, \mathbf{x} \rangle$. The master key holder, with knowledge of Δ, can compute $k = \langle \mathbf{z_0}, \mathbf{x} \rangle - \Delta \cdot \alpha = k'$, where $\alpha = \langle \mathbf{z}, \mathbf{x} \rangle$. This is enough to satisfy the correctness property of Definition 3 (Shiftable CPRFs). In particular, here the constraint predicate $C(\mathbf{x}, \alpha)$ is 0 if $\langle \mathbf{z}, \mathbf{x} \rangle - \alpha = 0$ and 1 otherwise.

Moving to the Ring Setting. We require instantiating the ShCPRF with a *small* ring $\mathcal{R}$ (e.g., $\mathcal{R} = \mathbb{Z}_6$) for efficiency purposes. However, to ensure each derived key is still at least λ-bits, we must extend the small ring to a sufficiently large ring $\mathcal{R}'$. As such, we replace the large field $\mathbb{F}$ with a large $\mathcal{R}' = \mathcal{R}^m$ where $m \geq \lambda$ to guarantee λ bits of security (we prove the security of this modification in the full version [29]). Doing so, however, makes the vectors $\mathbf{z_0}$ and $\mathbf{z_1}$ (now sampled in $(\mathcal{R}')^n$) better denoted as *matrices* from $\mathcal{R}^{m\times n}$. While this induces notational changes, the CPRF construction itself remains almost identical to the one of Servan-Schreiber. In particular, for a constraint $\mathbf{z} \in \mathcal{R}^n$, we now have CPRF keys $\mathbf{k}, \mathbf{k}' \in \mathcal{R}^m$ (as opposed to $k, k' \in \mathbb{F}$ above) derived for an input $\mathbf{x} \in \mathcal{R}^n$ as $\mathbf{k} = \mathbf{Z_0x} = \mathbf{Z_1x} + (\varDelta\mathbf{z}^\top)\mathbf{x}$ which is equal to $\mathbf{k}' = \mathbf{Z_1x}$, when the constraint $\langle \mathbf{z}, \mathbf{x}\rangle = 0 \in \mathcal{R}$. We present our ring-based ShCPRF framework in Fig. 1 and show security in Sect. 4.3 and the full version [29].

ShCPRF for Inner-Product Predicates

Public Parameters. Security parameter λ, finite ring $\mathcal{R}$ of order ℓ, integers m such that $m \geq \lambda$, vector length $n \geq 1$, and an RKA-secure PRF family $F\colon \mathcal{R}^m \times \mathcal{R}^n \to \mathcal{Y}$ for affine key-derivation functions.

```
ShCPRF.KeyGen($1^\lambda$):
  1 : $\mathbf{k_0} \xleftarrow{R} \mathcal{R}^m,\ \varDelta \xleftarrow{R} \mathcal{R}^m \setminus \{0\}$
  2 : $\mathbf{Z_0} \xleftarrow{R} \mathcal{R}^{m\times n}$
  3 : msk := $(\mathbf{k_0}, \mathbf{Z_0}, \varDelta)$

ShCPRF.Constrain(msk, $\mathbf{z}$):
  1 : parse msk = $(\mathbf{k_0}, \mathbf{Z_0}, \varDelta)$
  2 : $\mathbf{Z_1} \leftarrow \mathbf{Z_0} - \varDelta\mathbf{z}^\top$
  3 : return csk := $(\mathbf{k_0}, \mathbf{Z_1})$

ShCPRF.Eval(msk, $\mathbf{x}$, $\alpha$):
  1 : parse msk = $(\mathbf{k_0}, \mathbf{Z_0}, \varDelta)$
  2 : $\mathbf{k} \leftarrow \mathbf{k_0} + \mathbf{Z_0x} - \varDelta \cdot \alpha$
  3 : return $F_{\mathbf{k}}(\mathbf{x})$

ShCPRF.CEval(csk, $\mathbf{x}$):
  1 : parse csk := $(\mathbf{k_0}, \mathbf{Z_1})$
  2 : $\mathbf{k} \leftarrow \mathbf{k_0} + \mathbf{Z_1x}$
  3 : return $F_{\mathbf{k}}(\mathbf{x})$
```

Fig. 1. ShCPRF framework for inner-product predicates based on RKA-secure PRFs.

To state more exactly the special type of Shiftable CPRF we obtain, we have the following definition.

Definition 4 (Shiftable CPRFs for Inner-Product Predicates). *Let $\mathcal{R} = \mathcal{R}_\lambda$ be a finite ring. Let* ShCPRF *be a Shiftable CPRF with domain $\mathcal{X} = \mathcal{R}^n$ for an $n = n(\lambda)$, range $\mathcal{R}$, and finite set of shifts $\mathcal{S} = \mathcal{R}$ that supports constraints represented by a class of circuits $\{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$, such that $\mathcal{C}_\lambda = \{C_\mathbf{z} \colon \mathbf{z} \in \mathcal{R}^n\}$, where the $C_\mathbf{z}\colon \mathcal{X} \times \mathcal{S} \to \{0,1\}$ are given via*

$$(\mathbf{x}, \alpha) \mapsto \begin{cases} 0, & \textit{if } \langle \mathbf{z}, \mathbf{x}\rangle - \alpha = 0, \\ 1, & \textit{otherwise.} \end{cases}$$

Then we identify the constraint circuit $C_\mathbf{z}$ with $\mathbf{z}$, i.e. we just write that the constraint is a vector $\mathbf{z}$, and call ShCPRF *a (ring-based)* Shiftable CPRF for inner-product predicates *(with domain $\mathcal{R}^n$).*

4.3 Security Analysis

Here, we analyze the security of the ShCPRF framework using the proof template of Servan-Schreiber [61]. The proof follows the proof of [61, Theorem 2], however, we adapt it in several key locations to handle the shiftability property and operations in the ring $\mathcal{R}$.

Theorem 1. *If $\mathcal{F}$ is a family of RKA-secure pseudorandom functions with respect to affine related-key derivation functions Φ_{aff}, as defined in the full version, then Fig. 1 instantiated with $\mathcal{F}$ is a (1-key, selectively-secure) ShCPRF for inner-product constraint predicates.*

Proof. Deferred to the full version [29]. ■

5 PCFs for ListOT: Framework

In this section, we formalize our PCF for ListOT framework using the Shiftable CPRF framework from Sect. 4. As mentioned in Sect. 2, ListOT does *not* fulfill the definition of a "correlation" as defined by Boyle et al. [20]. Therefore, we cannot use existing definitions of a Pseudorandom *Correlation* Function (PCF), since the correlation is only partially defined. In particular, the problem is that the output of the receiver in ListOT has an additional lookup key i that depends directly on the wPRF key used to compute the pseudorandom bit b, which cannot be efficiently sampled given just the output of the sender. We sidestep these issues by adapting the standard definition of a PCF [19] to work with the "partial correlation" that is ListOT. In Sect. 5.1, we define the notion of a PCF for ListOT. Then, in Sect. 5.2, we describe our general framework for constructing a PCF for ListOT. Finally, in Sect. 5.3, we explain how a PCF for ListOT is used to instantiate QuietOT.

5.1 Defining PCFs for ListOT

Here, we give a formal definition of (weak) PCF for ListOT. For convenience, we use of the following *distribution of lists* notation.

Definition 5 (Distribution of Lists). *Let λ be a security parameter, $\mathcal{Y}$ be a finite set, and I be an arbitrary finite index set. We denote by $\mathcal{D}^{\text{list}}_{\mathcal{Y}}(I)$ the distribution that outputs a list $(v_i)_{i\in I}$, where each $v_i \xleftarrow{R} \mathcal{Y}$ is independently sampled at random.*

Definition 6 (Pseudorandom Correlation Function for ListOT). *Let λ be a security parameter and $\lambda \leq n = n(\lambda) \in \mathsf{poly}(\lambda)$ be an input length. A Pseudorandom Correlation Function (PCF) for ListOT with domain $\mathcal{X} = \mathcal{X}_\lambda$ is defined by a pair of algorithms $(\mathsf{PCF.KeyGen}, \mathsf{PCF.Eval})$ with the following functionality:*

- $\mathsf{PCF.KeyGen}(1^\lambda) \to (K_S, K_R)$. *Takes as input the security parameter* λ. *Outputs a pair of keys* (K_S, K_R).
- $\mathsf{PCF.Eval}(\sigma, K_\sigma, x) \to y_\sigma$. *Takes as input* $\sigma \in \{S, R\}$, *a key* K_σ, *and input* $x \in \mathcal{X}$. *Outputs a string* $y_\sigma \in \{0,1\}^*$, *where*
  - *if* $\sigma = S$ *then* $y_\sigma = (L_0, L_1)$ *for two lists* L_0, L_1, *and*
  - *if* $\sigma = R$ *then* $y_\sigma = (b, v, \alpha)$ *for a bit* $b \in \{0,1\}$, *a list entry* $v \in L_0 \cup L_1$, *and a lookup key* α.

We will use $\mathsf{PCF.EvalS}(K_S, x)$ *and* $\mathsf{PCF.EvalR}(K_R, x)$ *as shorthand for the* Eval *algorithm used by the sender and receiver, respectively. We leave any public parameters* PP *as an implicit input to all algorithms.*

A $\mathsf{PCF} = (\mathsf{PCF.KeyGen}, \mathsf{PCF.Eval})$ *is a* (weak) PCF for ListOT *with domain* $\mathcal{X} = \mathcal{X}_\lambda$, *if the following* correctness, sender security, *and* receiver security *properties hold. In each case, the adversary is given access to* $N(\lambda) \in \mathsf{poly}(\lambda)$ *samples.*

- ***Pseudorandomness.*** *For all efficient adversaries* $\mathcal{A}$, *and all* $N \in \mathsf{poly}(\lambda)$, *there exists a negligible function* negl *such that for all sufficiently large* λ,
$$\mathsf{Adv}^{\mathsf{pr}}_{\mathcal{A},N}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,1}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda),$$
where $\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,b}(\lambda)$, *for* $b \in \{0,1\}$, *is as defined in Fig. 2.*
- ***Correctness.*** *Moreover, we want that for any* $\lambda \in \mathbb{N}$ *the following probability is negligible in* λ*:*
$$\Pr\left[\begin{array}{l} (K_S, K_R) \leftarrow \mathsf{PCF.KeyGen}(1^\lambda), \\ x \overset{R}{\leftarrow} \mathcal{X}_\lambda, \\ (L_0, L_1) \leftarrow \mathsf{PCF.EvalS}(K_S, x), \\ (b, v, \alpha) \leftarrow \mathsf{PCF.EvalR}(K_R, x) \end{array} : \ v \neq L_b[\alpha]\right],$$
i.e., that the (relevant) entry v *is at position* α *of list* L_b *with a probability that is overwhelming in* λ.
- ***Sender Security.*** *For all efficient adversaries* $\mathcal{A}$, *there exists a negligible function* negl *such that for all sufficiently large* λ,
$$\mathsf{Adv}^{\mathsf{Ssec}}_{\mathcal{A},N}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{Ssec}}_{\mathcal{A},N,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{Ssec}}_{\mathcal{A},N,1}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda),$$
where $\mathsf{Exp}^{\mathsf{Ssec}}_{\mathcal{A},N,b}(\lambda)$, *for* $b \in \{0,1\}$, *is as defined in Fig. 3.*
- ***Receiver Security.*** *For all efficient adversaries* $\mathcal{A}$, *there exists a negligible function* negl *such that for all sufficiently large* λ,
$$\mathsf{Adv}^{\mathsf{Rsec}}_{\mathcal{A},N}(\lambda) = \left|\Pr[\mathsf{Exp}^{\mathsf{Rsec}}_{\mathcal{A},N,0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{Rsec}}_{\mathcal{A},N,1}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda),$$
where $\mathsf{Exp}^{\mathsf{Rsec}}_{\mathcal{A},N,b}(\lambda)$, *for* $b \in \{0,1\}$, *is as defined in Fig. 4.*

```
$\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,0}(\lambda)$:
$(K_S, K_R) \leftarrow \mathsf{PCF.KeyGen}(1^\lambda)$
for $i = 1$ to $N(\lambda)$:
    $x_i \overset{R}{\leftarrow} \mathcal{X}_\lambda$
    for $\sigma \in \{S, R\}$:
        $y^i_\sigma \leftarrow \mathsf{PCF.Eval}(\sigma, K_\sigma, x_i)$
    parse $y^i_S = (L^i_0, L^i_1)$, $y^i_R = (b_i, v_i, \alpha_i)$
$b' \leftarrow \mathcal{A}(1^\lambda, (x_i, L^i_0, L^i_1, b_i)_{i \in [N(\lambda)]})$
return $b'$
```

```
$\mathsf{Exp}^{\mathsf{pr}}_{\mathcal{A},N,1}(\lambda)$:
for $i = 1$ to $N(\lambda)$:
    $x_i \overset{R}{\leftarrow} \mathcal{X}_\lambda$
    $L^i_0 \overset{R}{\leftarrow} \mathcal{D}^{\mathsf{list}}_{\mathcal{Y}}(S_0)$, $L^i_1 \overset{R}{\leftarrow} \mathcal{D}^{\mathsf{list}}_{\mathcal{Y}}(S_1)$
    $b_i \overset{R}{\leftarrow} \{0, 1\}$
$b' \leftarrow \mathcal{A}(1^\lambda, (x_i, L^i_0, L^i_1, b_i)_{i \in [N(\lambda)]})$
return $b'$
```

Fig. 2. (Partially) Pseudorandom outputs of a PCF for ListOT. The distribution $\mathcal{D}^{\mathsf{list}}_{\mathcal{Y}}(\cdot)$ is defined in Definition 5.

```
$\mathsf{Exp}^{\mathsf{Ssec}}_{\mathcal{A},N,0}(\lambda)$:
$(K_S, K_R) \leftarrow \mathsf{PCF.KeyGen}(1^\lambda)$
for $i = 1$ to $N(\lambda)$:
    $x_i \overset{R}{\leftarrow} \mathcal{X}_\lambda$
    $(L^i_0, L^i_1) \leftarrow \mathsf{PCF.EvalS}(K_S, x_i)$
$b' \leftarrow \mathcal{A}(1^\lambda, K_R, (x_i, L^i_0, L^i_1)_{i \in [N(\lambda)]})$
return $b'$
```

```
$\mathsf{Exp}^{\mathsf{Ssec}}_{\mathcal{A},N,1}(\lambda)$:
$(K_S, K_R) \leftarrow \mathsf{PCF.KeyGen}(1^\lambda)$
for $i = 1$ to $N(\lambda)$:
    $x_i \overset{R}{\leftarrow} \mathcal{X}_\lambda$
    $(b_i, v_i, \alpha_i) \leftarrow \mathsf{PCF.EvalR}(K_R, x_i)$
    $L^i_0 \overset{R}{\leftarrow} \mathcal{D}^{\mathsf{list}}_{\mathcal{Y}}(S_0)$, $L^i_1 \overset{R}{\leftarrow} \mathcal{D}^{\mathsf{list}}_{\mathcal{Y}}(S_1)$
    Set $L^i_{b_i}[\alpha_i] := v_i$
$b' \leftarrow \mathcal{A}(1^\lambda, K_R, (x_i, L^i_0, L^i_1)_{i \in [N(\lambda)]})$
return $b'$
```

Fig. 3. Sender security game of a PCF for ListOT. The distribution $\mathcal{D}^{\mathsf{list}}_{\mathcal{Y}}(\cdot)$ is defined in Definition 5.

5.2 Framework: PCF for ListOT from IPM-wPRFs

In Fig. 5, we describe the framework for constructing a PCF for ListOT by combining a Shiftable CPRF with any IPM-wPRF.

Theorem 2. *Let $n = n(\lambda) \in \mathsf{poly}(\lambda)$ and $\mathcal{R}$ be a finite ring of order q, with extension parameter $m \geq \lambda$. Let $\mathsf{ShCPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ be a shiftable CPRF for inner-product predicates with domain $\mathcal{R}^n$, and $f \colon \mathcal{R}^n \times \mathcal{R}^n \rightarrow \{0, 1\}$ a weak PRF family for inner-product membership with partitioning $S_0 \cup S_1 = \mathcal{R}$. Then, $\mathsf{PCF} = (\mathsf{KeyGen}, \mathsf{Eval})$ from Fig. 5 is a PCF for ListOT.*

Proof. Deferred to the full version. ■

5.3 Realizing QuietOT from a PCF for ListOT

To generate random OT correlations, the sender and receiver use the PCF for ListOT to generate pseudorandom ListOT instances. We use the template of Beaver [12] for converting a random ListOT instance into a chosen-bit OT protocol. We describe this transformation in Fig. 6.

```
$\mathsf{Exp}^{\mathsf{Rsec}}_{\mathcal{A},N,0}(\lambda)$:
$(K_S, K_R) \leftarrow \mathsf{PCF.KeyGen}(1^\lambda)$
for $i = 1$ to $N(\lambda)$:
    $x_i \xleftarrow{R} \mathcal{X}_\lambda$
    $(b_i, v_i, \alpha_i) \leftarrow \mathsf{PCF.EvalR}(K_R, x_i)$
$b' \leftarrow \mathcal{A}(1^\lambda, K_S, (x_i, b_i)_{i\in[N(\lambda)]})$
return $b'$
```

```
$\mathsf{Exp}^{\mathsf{Rsec}}_{\mathcal{A},N,1}(\lambda)$:
$(K_S, K_R) \leftarrow \mathsf{PCF.KeyGen}(1^\lambda)$
for $i = 1$ to $N(\lambda)$:
    $x_i \xleftarrow{R} \mathcal{X}_\lambda$
    $b_i \xleftarrow{R} \{0,1\}$
$b' \leftarrow \mathcal{A}(1^\lambda, K_S, (x_i, b_i)_{i\in[N(\lambda)]})$
return $b'$
```

Fig. 4. Receiver security game of a PCF for ListOT.

Proposition 1. *Let* $\mathsf{PCF} = (\mathsf{PCF.KeyGen}, \mathsf{PCF.Eval})$ *be a PCF for ListOT. Then the protocol given in Fig. 6 securely realizes the OT functionality.*

Proof. The OT functionality is defined in the full version. By the pseudorandomness property of the PCF for ListOT we have that L_0 and L_1 are pseudorandom lists and b' is a pseudorandom bit if x (input to the PCF) is uniformly random. For an arbitrary choice bit $b \in \{0,1\}$ we consider the two possible cases to prove correctness.

- Case 1: $b = 0$. In this case, $c = b'$ and so $L'_0 = L_{b'} \oplus m_0$ and $L'_1 = L_{1-b'} \oplus m_1$. It then follows that the receiver outputs $(L'_0[\alpha] \oplus m_0) \oplus v$, which equals m_0 by the correctness property of the PCF.
- Case 2: $b = 1$. In this case, $c = 1 - b'$ and so $L'_0 = L_{1-b'} \oplus m_0$ and $L'_1 = L_{b'} \oplus m_1$. It then follows that the receiver outputs $m_1 = (L'_1[\alpha] \oplus m_1) \oplus v$, since we have $v = L_{b'}[\alpha] = L'_1[\alpha]$ by the correctness property of the PCF.

Note that in both cases, the equality holds with overwhelming probability, because correctness of the PCF holds with overwhelming probability ((Definition 6).

Sender security follows directly from the sender security of the PCF which guarantees that (1) the receiver only obtains $L_{b'}[\alpha]$ and (2) all other values in both lists are pseudorandom from the viewpoint of the receiver and therefore guarantees that the receiver only obtains m_b.

Receiver security follows from the fact that b' is a pseudorandom bit (by receiver security of the PCF) and therefore a pseudorandom mask for the receiver's choice bit b. Therefore, the sender learns nothing. ■

6 PCFs for ListOT: Instantiations

In this section, we instantiate the framework from Sect. 5 using either the BIPSW or GAR IPM-wPRF candidate, coupled with the "RKA-PRF" $F_k(x) := H(k, x)$ for a hash function H modeled as a random oracle. For the sake of completeness, we prove in the full version [29] that F_k is indeed an RKA-PRF for all affine relations $x \mapsto \alpha x + \beta$ with $\alpha \in \mathcal{R}^*$ and $\beta \in \mathcal{R}^m$, as long as $\min_\alpha(|\alpha \cdot \mathcal{R}^m|) \geq 2^\lambda$.

PCF for ListOT

Public Parameters (PP).

- Key length $n = n(\lambda) \in \mathsf{poly}(\lambda)$.
- Finite ring $\mathcal{R}$ of order q, with extension parameter $m \geq \lambda$.
- IPM-wPRF family $f : \mathcal{R}^n \times \mathcal{R}^n \to \{0, 1\}$ with partitioning $S_0 \cup S_1 = \mathcal{R}$.
- An injective input mapping $\mathsf{map}\colon \mathcal{X}_\lambda \to \mathcal{R}^n$.
- A ShCPRF $\mathsf{ShCPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ with domain $\mathcal{R}^n$

```
PCF.KeyGen($1^\lambda$).
1: $\mathsf{msk} \leftarrow \mathsf{ShCPRF.KeyGen}(1^\lambda)$
2: $\mathbf{z} \overset{R}{\leftarrow} \mathcal{R}^n$. ▷ IPM-wPRF key
   ▷ Note that $\mathbf{z}$ can also be sampled from a non-uniform distribution over $\mathcal{R}^n$.
3: $\mathsf{csk} \leftarrow \mathsf{ShCPRF.Constrain}(\mathsf{msk}, \mathbf{z})$
4: $K_S := \mathsf{msk}$, $K_R := (\mathsf{csk}, \mathbf{z})$.
5: return $(K_S, K_R)$

PCF.EvalS($K_S, x$).
1: parse $K_S = \mathsf{msk}$.
2: $\mathbf{x} \leftarrow \mathsf{map}(x)$.
3: foreach $b \in \{0, 1\}$ and $\alpha \in S_b$:
    1: $y \leftarrow \mathsf{ShCPRF.Eval}(\mathsf{msk}, \mathbf{x}, \alpha)$.
    2: $L_b[\alpha] = y$.
4: return $(L_0, L_1)$.

PCF.EvalR($K_R, x$).
1: parse $K_R = (\mathsf{csk}, \mathbf{z})$.
2: $\mathbf{x} \leftarrow \mathsf{map}(x)$.
3: $v \leftarrow \mathsf{ShCPRF.CEval}(\mathsf{csk}, \mathbf{x})$
4: $b \leftarrow f_{\mathbf{z}}(\mathbf{x})$, $\alpha \leftarrow \langle \mathbf{z}, \mathbf{x} \rangle \in \mathcal{R}$.
   ▷ Note that $v = L_b[\alpha]$ in the sender-computed list.
5: return $(b, v, \alpha)$.
```

Fig. 5. Framework for a PCF for ListOT from any ShCPRF and IPM-wPRF.

Looking ahead, both our instantiations will satisfy $\min_\alpha(|\alpha \cdot \mathcal{R}^m|) = 2^m$, and we will therefore set $m = \lambda$.

These two instantiations result in our concretely efficient constructions (see Sect. 7). We also describe other instantiations from different assumptions (in particular, replacing the random oracle using an RKA-secure PRF), which have interesting theoretical implications but do not currently result in concretely efficient constructions.

QuietOT from a PCF for ListOT

Sender(K_S, m_0, m_1, x)		**Receiver**(K_R, b, x)
$(L_0, L_1) \leftarrow \mathsf{PCF.EvalS}(K_S, x)$		$(b', v, \alpha) \leftarrow \mathsf{PCF.EvalR}(K_R, x)$
	$\xleftarrow{c = b \oplus b'}$	
$L'_0 \leftarrow L_c \oplus m_0$		
$L'_1 \leftarrow L_{1-c} \oplus m_1$	$\xrightarrow{L'_0, L'_1}$	
Output $\perp$		Output $L'_b[\alpha] \oplus v$

Fig. 6. QuietOT using a (weak) PCF for ListOT. Note that x (input to the PCF known by both the sender and receiver) is uniformly random, as specified in Definition 6.

6.1 BIPSW IPM-wPRF Instantiation

Our main instantiation is based on the BIPSW wPRF candidate which can be easily viewed as an IPM-wPRF. For a key $\mathbf{z} \in \mathbb{Z}_6^n$ with $n = n(\lambda) \in \mathsf{poly}(\lambda)$, and $\mathbf{x} \in \mathbb{Z}_6^n$, the BIPSW wPRF is defined as: $f_{\mathbf{z}}(\mathbf{x}) = \lfloor \langle \mathbf{z}, \mathbf{x} \rangle \mod 6 \rceil_2$, where $\lfloor \alpha \rceil_2 = 0$ for all $\alpha \in \{0, 1, 2\}$ and $\lfloor \alpha \rceil_2 = 1$ for all $\alpha \in \{3, 4, 5\}$. For a partitioning of $\mathbb{Z}_6$ consisting of $S_0 := \{0, 1, 2\}$ and $S_1 := \{3, 4, 5\}$, we get that $\langle \mathbf{z}, \mathbf{x} \rangle \mod 6 \in S_b \iff f_{\mathbf{z}}(\mathbf{x}) = b$.

We instantiate the framework using the ring $\mathcal{R} = \mathbb{Z}_6$ and set $m \geq \lambda$ (see the full version [29]). We interpret $\{0, 1\}$ as elements of $\mathbb{Z}_6$ in the natural way (mapping 0 to the additive identity and 1 to the multiplicative identity of $\mathbb{Z}_6$) and define map to be the canonical embedding from $\{0, 1\}^n$ to $\mathbb{Z}_6^n$. The full construction is presented in Fig. 7 and closely follows the general framework from Fig. 5 with the exception that we use the specific ShCPRF construction of Fig. 1, and fix the RKA-secure PRF to be the random oracle H, and additionally explicitly work over the ring $\mathbb{Z}_6$.

6.2 GAR IPM-wPRF Instantiation

Unlike for the BIPSW wPRF, converting the GAR wPRF into an IPM-wPRF is more challenging. For completeness, we describe how Bui et al. [25] express the evaluation function as an IPM and discuss concrete parameters that we use for our instantiation, which differ from the parameters used to instantiate BCMPR.

The GAR Construction. In a nutshell, the GAR construction (when instantiated with the XOR-MAJ predicate [3,31]) has a key $K \in \{0, 1\}^n$ and takes as input a string x that is parsed as a tuple of disjoint *sets* $(X_{\mathsf{xor}}, X_{\mathsf{maj}}) \subset [n]^2$ such

PCF for ListOT from the BIPSW IPM-wPRF

Parameters.

– Integers $n = n(\lambda) \in \mathsf{poly}(\lambda)$, $m \geq \lambda$ and domain $\mathcal{X}_\lambda = \{0,1\}^n$.
– BIPSW IPM-wPRF family $f : \mathbb{Z}_6^n \times \mathbb{Z}_6^n \to \{0,1\}$ with partitioning:
$$S_0 := \{\alpha \in \mathbb{Z}_6 \mid \alpha < 3\} \text{ and } S_1 := \mathbb{Z}_6 \setminus S_0.$$
– Input mapping $\mathsf{map}\colon \mathcal{X}_\lambda \to \mathcal{R}^n$ is the canonical embedding of $\{0,1\}^n$ in $\mathbb{Z}_6^n$.
– The ShCPRF $\mathsf{ShCPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ from Figure 1, where the RKA-secure PRF family $F\colon \mathcal{R}^m \times \mathcal{R}^n \to \mathcal{Y}$ is instatiated by a random oracle H, cf. the full version.

Construction.
($\mathsf{PCF.KeyGen}$, $\mathsf{PCF.EvalR}$, $\mathsf{PCF.EvalS}$) are identical to Figure 5 using $\mathcal{R} = \mathbb{Z}_6$.

Fig. 7. PCF for ListOT from the BIPSW IPM-wPRF

that $|X_{\mathsf{xor}}| = k$ and $|X_{\mathsf{maj}}| = \ell$, for integers $k = k(\lambda), \ell = \ell(\lambda)$. The evaluation of f_K is then computed as: $(\bigoplus_{i \in X_{\mathsf{xor}}} K[i]) \oplus \mathsf{MAJ}((K[j])_{j \in X_{\mathsf{maj}}})$, where MAJ outputs the majority bit.

The GAR construction as an IPM-wPRF. Converting this evaluation into an inner-product membership can be done as follows. View the evaluation as two separate components: an XOR component and a MAJ component. For each index $i \in X_{\mathsf{xor}}$, let $\mathbf{e}_i$ be the one-hot vector of length n with 1 in its i-th coordinate. First, interpret K as $\mathbf{z}_{\mathsf{xor}} \in \mathbb{Z}_2^n$ and as $\mathbf{z}_{\mathsf{maj}} \in \mathbb{Z}_\ell^n$ by mapping 0 to $0 \in \mathbb{Z}_2$ (resp. $\mathbb{Z}_\ell$) and 1 to $1 \in \mathbb{Z}_2$ (resp. $\mathbb{Z}_\ell$)). Then, compute $v_{\mathsf{xor}} = \sum_{i \in X_{\mathsf{xor}}} \langle \mathbf{z}_{\mathsf{xor}}, \mathbf{e}_i \rangle$. Similarly, for each index $j \in X_{\mathsf{maj}}$, let $\mathbf{e}_j$ be the corresponding one-hot vector. Then, compute $v_{\mathsf{maj}} = \sum_{j \in X_{\mathsf{maj}}} \langle \mathbf{z}_{\mathsf{maj}}, \mathbf{e}_j \rangle$. Observe that $v_{\mathsf{maj}} \geq \lceil \frac{\ell}{2} \rceil \iff \mathsf{MAJ}((\mathbf{z}_{\mathsf{maj}}[j])_{j \in X_{\mathsf{maj}}}) = 1$. We define $\mathcal{R} = \mathbb{Z}_2 \times \mathbb{Z}_\ell$, which intuitively allows for computing the "XOR" and "MAJ" components in separate subrings. Therefore, we can view f as an IPM-wPRF with partition:

$$S_0 = \left\{(u,v) \in \mathcal{R} \mid (u = 0 \land v > \lfloor \tfrac{\ell}{2} \rfloor) \lor (u = 1 \land v \leq \lfloor \tfrac{\ell}{2} \rfloor)\right\} \text{ and } S_1 = \mathcal{R} \setminus S_0.$$

Parameters. We follow the parameter selection process of Bui et al. [25], which builds upon the state-of-the-art cryptanalysis of Goldreich's PRG from [3,31,63,64]. To achieve λ bits of security with a key of length $n = \lambda^\delta$ and a bound n^{1+e} on the number of queries, the analysis of Bui et al. [25] suggests to use the XOR-MAJ predicate with $\ell_1 = 2 \cdot e + 1$ terms in the XOR, and $\ell_2 = (2\delta/(\delta - 1)) \cdot e + 1$ terms in the MAJ. Concretely, we set $e = 2$ to get a stretch n^3 (looking ahead, we will choose $n = 2^{11}$, hence this corresponds to generating up to 2^{33} OTs) and $\delta = 7/5$ (hence $\delta/(\delta - 1) = 7/2$). This implies that we can set $\ell_1 = 5$ and $\ell_2 = 15$. With these parameters, we must set $\ell = \ell_2 + 1 = 16$ to ensure no wraparound when computing the MAJ predicate on the sum modulo ℓ of the ℓ_2 entries in the corresponding subset. While this analysis indicates that a seed size of $n \geq 128^{\delta} = 892$ suffices, we set $n = 2048$ which allows us to more efficiently parse uniformly random inputs x into the index sets X_{xor} and X_{maj}, and generate a larger number $\lambda^{\delta} = 2^{33}$ of oblivious transfers. This results in an extremely conservative parameter set: The estimated bit security of this parameter set, using the state-of-the-art cryptanalysis [3,31,63,64], is 2^{232}.

PCF for ListOT from the GAR IPM-wPRF

Parameters.

- Integers $n = n(\lambda) \in \mathsf{poly}(\lambda)$ and $m \geq \lambda$ and $\mathcal{R} = \mathbb{Z}_2 \times \mathbb{Z}_\ell$.
- GAR IPM-wPRF family $f : \{0,1\}^n \times \{0,1\}^w \to \{0,1\}$ with parameters $w_0, w_1, w = w(w_0, w_1)$, where $w_0, w_1, w \in \mathbb{N}$, and partitioning:
  $S_0 = \left\{(u, v) \in \mathcal{R} \mid (u = 0 \wedge v > \left\lfloor \frac{\ell}{2} \right\rfloor) \vee (u = 1 \wedge v \leq \left\lfloor \frac{\ell}{2} \right\rfloor)\right\}$ and $S_1 = \mathcal{R} \setminus S_0$.
- Input mapping $\mathsf{map}\colon \{0,1\}^{\mathsf{poly}(n)} \to \mathcal{R}^n$ where each uniformly random input $x \in \{0,1\}^{\mathsf{poly}(n)}$ is interpreted as a tuple $(x_{\mathsf{xor}}, x_{\mathsf{maj}}) \in \{0,1\}^n \times \{0,1\}^n$ subject to $\mathcal{HW}(x_{\mathsf{xor}}) = w_0$ and $\mathcal{HW}(x_{\mathsf{maj}}) = w_1$, where $\mathcal{HW}(\cdot)$ denotes the Hamming weight of the input. $(x_{\mathsf{xor}}, x_{\mathsf{maj}})$ is interpreted as $(\mathbf{x}_{\mathsf{xor}}, \mathbf{x}_{\mathsf{maj}}) \in \mathcal{R}^n$ in the natural way (by interpreting 0 and 1 as elements of $\mathcal{R}$).
- The ShCPRF $\mathsf{ShCPRF} = (\mathsf{KeyGen}, \mathsf{Eval}, \mathsf{Constrain}, \mathsf{CEval})$ from Figure 1, where the RKA-secure PRF family $F\colon \mathcal{R}^m \times \mathcal{R}^n \to \mathcal{Y}$ is instatiated by a random oracle H, cf. the full version.

Construction.
($\mathsf{PCF.KeyGen}$, $\mathsf{PCF.EvalR}$, $\mathsf{PCF.EvalS}$) are as described in Figure 5 using the ring $\mathcal{R} = \mathbb{Z}_2 \times \mathbb{Z}_\ell$ and input mapping map defined above.

Fig. 8. PCF for ListOT from the GAR IPM-wPRF

6.3 Other Instantiations

While we focus on the BIPSW and GAR IPM-wPRF constructions when instantiating our framework, several other instantiations are possible. First, we can instantiate the framework using a different IPM-wPRF candidate. While BIPSW and GAR appear to be the most efficient candidates to fit the IPM-wPRF template, future wPRF candidates or improved parameters for the LWR wPRF resulting from tighter reductions for the LWR problem, could lead to new instantiations. For example, with the VDLPN wPRF candidate [19] (which can be cast as an

IPM-wPRF [25]) and whose concrete security is beginning to be analyzed [30], we could potentially have an additional practical instantiation.

Additionally, our framework is not restricted to the random oracle model (albeit, assuming a random oracle can lead to the most practical instantiations, as is the case for other OT extension protocols). As with prior OT extension protocols, we can replace the random oracle with a suitable correlation-robust hash function [47]. However, we can even go one step further. Note that because the security relies on the security of the ShCPRF, and the CPRF construction of Servan-Schreiber [61] relies only any suitable RKA-secure PRF (a property inherited by our construction of shiftable CPRFs), we can instantiate it using other assumptions such as DDH, DCR, or VDLPN. We discuss these (currently purely theoretical) instantiations in the full version [29].

7 Implementation and Evaluation

Implementation. We implement QuietOT in C with roughly 1200 lines of code for the BIPSW and GAR implementations combined. Our implementation is open source.[10] We additionally implement the BCMPR silent OT protocol in C in roughly 600 lines of code using the P-256 elliptic curve implementation available in the OpenSSL library [62]. For OSY, we estimate their runtime by benchmarking the dominant cost of their construction: computing $\lambda = 128$ modular exponentiations in a 3200-bit RSA group. To heuristically instantiate the random oracle $H(\cdot)$, we use fixed-key AES, operating with 128-bit inputs, and truncate the output to a single bit (such an instantiation is shown to be correlation-robust in the idea cipher model [43]). To generate pseudorandom inputs for the PCF, we stretch a short seed (common to both the sender and receiver) using AES in CTR mode. Our implementations make use of several optimizations, which are described further in Sect. 7.1. We use the state-of-the-art implementation of existing OT extension protocols (IKNP, SoftSpoken, RRT) available in libOTe [58] to compare to other OT extension protocols. In order to provide a fairer comparison to existing OT extension protocols, we do *not* include the base OT costs required in SoftSpoken and IKNP.

Environment. We run our benchmarks on an AWS `c5.metal` and `t2.small` instances, and on an Apple M1 Pro laptop computer, using a single core. Because network latency and bandwidth can fluctuate leading to high variance, our benchmarks take into account only the processing time required by the sender and receiver. We compare the network overhead between each protocol using the “bits/OT” measure, which provides an objective and consistent comparison between protocols, avoiding network-specific or implementation differences.[11]

Parameters. We fix the security parameter $\lambda = 128$. For BIPSW, we set $n = 768$ and pre-compute inner products with CPRF keys in blocks of 16 bits

[10] https://github.com/sachaservan/QuietOT.

[11] The libOTe implementation is evaluated on `localhost`, and therefore is somewhat limited by the kernel when transferring data making IKNP slower than SoftSpoken.

(see Sect. 7.1). We operate over the ring $\mathbb{Z}_6$, which allows us to use CRT decomposition and pack 128 elements of $\mathbb{Z}_2$ into one machine word. For GAR, we set $n = 2048$, $\ell_1 = 5$ and $\ell_2 = 15$. This allows us to work over the ring $\mathcal{R} = \mathbb{Z}_2 \times \mathbb{Z}_{16}$. The choice of $n = 2048$ is very conservative but allows us to sample indices in $\{1, \ldots, 2048\}$ efficiently without rejection sampling. In turn, this improves concrete performance by allowing us to efficiently generate inputs for the wPRF (all we require is checking that the sampled set of random indices consist of distinct elements). SoftSpoken has a tunable tradeoff between communication and computational efficiency parameterized by k. For a given k SoftSpoken requires λ/k communication but increases computation by a factor of $2^k/k$. Small values of k (e.g., $k = 4$) provide a good tradeoff in practice, resulting in 32 bits/OT at an increase of $4\times$ in computation.

Communication Costs and Comparison. QuietOT with BIPSW as the IPM-wPRF requires 7 bits of communication per chosen-bit OT. For random choice bits, communication is only 6 bits since the receiver does not need to send its masked bit. Moreover, for random OT (when the sender inputs are also random), the messages m_0 and m_1 can be set to the first elements of L_0 and L_1, respectively, reducing communication to $|S_0| + |S_1| - 2$ (or 4 bits when using the BIPSW IPM-wPRF). QuietOT with GAR as the IPM-wPRF requires 33 bits of communication per chosen-bit OT. However, the same logic above reduces communication to 32 bits/OT when the choice bit is random and 30 bits/OT for random OT. QuietOT beats SoftSpoken on communication (for reasonable choices of k) when instantiated with BIPSW and remains on-par with SoftSpoken in terms of communication when instantiated with GAR. Silent OT protocols (i.e., RRT, BCMPR, OSY) have an optimal 3 bits/OT of communication and 2 bits/OT when the receiver's choice bit is random. This makes QuietOT roughly 2-10$\times$ worse in terms of communication when compared to Silent OT (Table 2).

Computational Costs and Comparison. The state-of-the-art OT extension protocol is SoftSpoken. To provide an apples-to-apples comparison of the computational costs while fixing the communication overhead in SoftSpoken, we could set $k = 18$ and $k = 4$ in SoftSpoken, leading to 7.1 bits/OT and 32 bits/OT, respectively. However, SoftSpoken becomes very inefficient with large k which does not make the comparison fair when QuietOT is instantiated using BIPSW. Comparing to SoftSpoken with small k and QuietOT (when instantiated with either BIPSW or GAR) shows that QuietOT is roughly one to two orders of magnitude slower. However, we stress that SoftSpoken benefits a lot more from advanced hardware instructions than QuietOT, potentially making QuietOT outshine SoftSpoken on weak(er) devices. This is evidenced by QuietOT outperforming the SoftSpoken implementation on the M1 (where AVX512 is not available). Unfortunately, since the libOTe implementation is not optimized for performance when AVX is disabled, performing a head-to-head comparison difficult. Comparing QuietOT to BCMPR (state-of-the-art *public-key* OT protocol) shows that QuietOT is up to 100$\times$ faster in terms of computation while only increasing communication by a few bits.

Table 2. OTs per second on a single core generated by the sender. Note that libOTe is not optimized for M1 since the AVX instructions are not available on M1 processors, hence we report these numbers in gray. The GAR IPM-wPRF cannot benefit from AVX due to limited bit-slicing opportunities. Setup costs are excluded.

	$\lvert pk_{sender} \rvert$	$\lvert pk_{receiver} \rvert$	**OT/s** (M1 Pro)	**OT/s** (c5.metal)	**OT/s** (t2.small)	**Bits/OT**
IKNP			2,592,000	34,174,000	12,264,000	128
SoftSpoken ($k = 2$)			2,732,000	52,676,000	33,121,000	64
SoftSpoken ($k = 4$)			1,636,000	44,443,000	27,504,000	32
SoftSpoken ($k = 8$)			249,000	9,500,000	5,891,000	16
SoftSpoken ($k = 16$)			2,000	76,000	49,000	8
RRT			1,230,000	6,856,000	2,492,000	3
OSY	50 kB	1 kB	0.6	0.5	0.3	3
BCMPR (BIPSW)	63 kB	72 kB	15,000	12,000	8,000	3
BCMPR (GAR)	33 kB	38 kB	21,000	17,000	11,000	3
QuietOT (BIPSW)	5.4 MB	84 MB	1,165,000	561,000	362,000	7
with AVX512 support			N/A	1,265,000	N/A	7
QuietOT (GAR)	5.6 MB	88 KB	1,198,000	526,000	336,000	33

Public Key Size. Our public key setup has public keys that are roughly 20 to 60 times larger compared to the public keys in BCMPR and OSY. This is primarily due to the parameters required for the RingLWE assumption (see the full version [29]). However, as a consequence, we obtain plausible post-quantum security. In practical terms, however, the average web page size is roughly 2 MB as of 2023 [46], making the overall key size very reasonable for use on the Internet. Additionally, we note that this is the *sender*'s public key size—the receiver's public key in our construction is only around 90kB, which could potentially be beneficial to some applications.

7.1 Optimizations in the Implementation

Our implementation makes use of several optimizations which we detail here.

Optimizing the BIPSW Instantiation. We make several observations allowing us to concretely optimize the BIPSW instantiation from Fig. 7.

Working Over the CRT Decomposition.. All computations over $\mathcal{R} = \mathbb{Z}_6$ can we computed over the CRT decomposition $\mathbb{Z}_2 \times \mathbb{Z}_3 \simeq \mathbb{Z}_6$. This enables applying the following two optimizations:

– *Bit-Sliced Arithmetic in .*$\mathbb{Z}_2$. We can pack the m elements of $\mathbb{Z}_2$ into $\lfloor m/64 \rfloor$ machine words (on 64-bit word processors). This allows parallelizing the $\mathbb{Z}_2$ component of all operations over $\mathcal{R}^m$ by using a single machine instruction for each batch of $\lfloor m/64 \rfloor$ elements of $\mathcal{R}^m$.

- *Bit-Sliced Arithmetic in .$\mathbb{Z}_3$.* While we can also pack the $\mathbb{Z}_3$ elements into $\lfloor 2m/64 \rfloor$ machine words (using two bits for each element of $\mathbb{Z}_3$ on a 64-bit machine), such a packing requires computing operations in $\mathbb{Z}_3$ over the bit-sliced representation. Fortunately, this very problem was explored in WAVE [7, Appendix B.1], where they provide an efficient bit-sliced representation for computing fast bit-sliced arithmetic over $\mathbb{Z}_3$. In particular, each arithmetic operation in $\mathbb{Z}_3$ requires using only 7 machine instructions.

Our Implementation in C. Concretely, we pack ring elements into `uint128_t` types in C (which represent two 64-bit machine words). This allows us to pack the 128 $\mathbb{Z}_2$ elements into one `uint128_t` type and pack the high and low order bits of the 128 $\mathbb{Z}_3$ elements into two `uint128_t` types. We can then perform bit-sliced arithmetic over this packed representation.

Preprocessing Inner Products. A separate optimization, which we find also applies to the construction of BCMPR (when instantiated with the BIPSW IPM-wPRF), is to preprocess the inner products over small chunks of the key. When using the BIPSW wPRF, each matrix-vector product $\mathbf{Z_0 x}$ can be equivalently written as a sum of smaller matrix-vector products. More precisely, let $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_{n/t})$ such that $|\mathbf{x}_i| \in \{0,1\}^t$ (we assume that t divides n) and let $\mathbf{Z_0} = (\mathbf{Z_0}_1, \ldots, \mathbf{Z_0}_{n/t})$. Then, by preprocessing all possible 2^t matrix-vector products associated with the i-th column block matrix $\mathbf{Z_0}_i$ and storing the results, we can efficiently look up the result for any input block $\mathbf{x}_i$, saving a factor t in computation at a cost of 2^t in extra storage.

Optimizing the GAR Instantiation Unfortunately, we find fewer ways to optimize the GAR instantation compared to our BIPSW instantiation. In particular, the GAR instantiation does not benefit from preprocessing opportunities that we identify for our BIPSW instantiation. However, we note that we can still take advantage of bit-sliced operations to compute the m operations over $\mathbb{Z}_2$ in parallel. Performing bit-sliced arithmetic over $\mathbb{Z}_{16}$ (when $\ell = 16$), in contrast, is more challenging.

Fast Arithmetic Over $\mathbb{Z}_{16}$. We found it to be more efficient to *not* pack the $\mathbb{Z}_{16}$ elements and instead just use one byte for each of the 128 elements of $\mathbb{Z}_{16}^{128}$. This allows us to sum modulo 16 by first computing the sum over the integers and then using a bit-mask to reduce modulo 16. In particular, we can sum two elements of $\mathbb{Z}_{16}$ via integer addition (summing two bytes) followed by a bit-wise AND with `0b00001111`, which zeroes-out the carry bit. This optimization makes summation modulo 16 fast, mitigating the impact of not being able to perform bit-sliced arithmetic for the $\mathbb{Z}_{16}$ elements.

General Optimization: Compressing Hash Inputs via Universal Hashing. In the ring element representations of both the BIPSW and GAR instantiations, we end up with a tightly packed representation of the $\mathbb{Z}_2$ elements but only a "loosely-packed" representation of the $\mathbb{Z}_3$ elements (resp. $\mathbb{Z}_{16}$ elements). The naive approach would be to feed the entire bit-string representation of the ring elements (the ShCPRF key) and input $\mathbf{x}$ into the random oracle, which is heuristically instantiated using fixed-key AES [43]. However, this would require breaking up the input string into blocks of 128 bits and then xoring all the resulting AES outputs together. While this solution does not introduce noticeable slowdowns for the BIPSW instantiation, it is not ideal for the GAR instantiation. For the GAR implementation, this approach would require packing all the elements of $\mathbb{Z}_{16}^{128}$ into four AES blocks which is inefficient since the packing itself is slow (recall that each element is represented as a byte for fast arithmetic operations). However, we additionally need to pack the ShCPRF input $\mathbf{x}$, which would lead to even more overheads. To avoid these inefficiencies, we instead choose to compress the representation of the ring elements and input $\mathbf{x}$ into tightly packed λ-bit strings by using a universal hash, which acts as a randomness extractor for the input to the random oracle. Specifically, we can make use of the leftover hash lemma [44] to extract $\lambda \approx 128$ bits from the representation of the ring elements. This allows us to then only perform one AES call, using the compressed 128-bit representation as input. We do the same for the ShCPRF input $\mathbf{x}$ and the $\mathbb{Z}_2^{128}$ block, leading to a total of three independent AES calls that we then truncate and xor together. The standard LHL bound requires $128 + 2\kappa$ bits of entropy to extract 128-bits that are statistically close to uniform in the worst-case [44], where κ is a statistical security parameter. However, the generalized LHL bound of Barak et al. [9] allows us to do better. Specifically, they prove that when extracting randomness to use as a key for a *weak* PRF (we assume that our ShCPRF takes uniformly random inputs and thus is a weak PRF) the LHL bound can be improved to $128 + \kappa$, which saves a factor of two in the entropy loss. Therefore, we can increase the ring dimension m to $128 + 64$ to ensure the universal hashing produces a near-uniform 128 bit key for the ShCPRF, with $\geq$ 64-bits of *statistical* security. As for the input $\mathbf{x}$, universal hashing provides 128-bits with even more statistical security since under our concrete parameter choice for GAR, we already have 308 bits of entropy in the input $\mathbf{x}$, which already give more than 64-bits of statistical security under the basic LHL bound. Finally, to further improve efficiency, we use an *almost*-universal (as opposed to perfectly universal) hash function, which results in faster implementations while only sacrificing a few bits of statistical security [9].

Acknowledgements. We thank the anonymous reviewers for helpful comments and pointing out a flaw in a previous version of our ShCPRF parameter choices. We thank Peter Rindal for help with running the libOTe library. We thank Sheela Devadas for helpful discussions on RingLWE. Geoffroy Couteau and Alexander Koch acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-20-CE39-0001 (project SCENE) and under the France 2030 ANR Project ANR-22-PECY-003 SecureCompute. This work is supported by ERC grant OBELiSC (101115790), and NSF grant 2330065.

References

1. M. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan. *Homomorphic Encryption Security Standard*. Tech. rep. HomomorphicEncryption.org, 2018. url: https://homomorphicencryption.org/wp-content/uploads/2018/11/HomomorphicEncryptionStandardv1.1.pdf
2. B. Applebaum, I. Damgård, Y. Ishai, M. Nielsen, and L. Zichron. "Secure arithmetic computation with constant computational overhead". In: *CRYPTO 2017*. Ed. by J. Katz and H. Shacham. LNCS 10401. Springer,2017, pp. 223–254. https://doi.org/10.1007/978-3-319-63688-7_8.
3. B. Applebaum and S. Lovett. "Algebraic attacks against random local functions and their countermeasures". In:*STOC 2016*. Ed. by D. Wichs and Y. Mansour. ACM, 2016, pp. 1087–1100. https://doi.org/10.1145/2897518.2897554.
4. B. Applebaum and P. Raykov. "Fast pseudorandom functions based on expander graphs". In: *TCC 2016-B*. Ed. by M. Hirt and A. D. Smith. LNCS 9985. Springer, 2016, pp. 27–56. https://doi.org/10.1007/978-3-662-53641-4_2.
5. G. Asharov, Y. Lindell, T. Schneider, and M. Zohner. "More efficient oblivious transfer and extensions for faster secure computation". In:*CCS 2013*. Ed. by A. Sadeghi, V. D. Gligor, and M. Yung. ACM, 2013, pp. 535–548. https://doi.org/10.1145/2508859.2516738.
6. N. Attrapadung, T. Matsuda, R. Nishimaki, S. Yamada, and T. Yamakawa. "Constrained PRFs for NC1 in traditional groups". In: *CRYPTO 2018*. Ed. by H. Shacham and A. Boldyreva. LNCS 10992. Springer, 2018, pp. 543–574. https://doi.org/10.1007/978-3-319-96881-0_19.
7. G. Banegas, K. Carrier, A. Chailloux, A. Couvreur, T. Debris-Alazard, P. Gaborit, P. Karpman, J. Loyer, R. Niederhagen, N. Sendrier, B. Smith, and J.-P. Tilich. WAVE: *Round 1Submission. Version 1*. 2023. url:https://wave-sign.org/wavedocumentation.pdf (visited on 01/21/2024).
8. A. Banerjee, C. Peikert, and A. Rosen. "Pseudorandom functions and lattices". In:*EUROCRYPT 2012*. Ed. by D. Pointcheval and T. Johansson. LNCS 7237. Springer, 2012, pp. 719–737. https://doi.org/10.1007/978-3-642-29011-4_42.
9. B. Barak, Y. Dodis, H. Krawczyk, O. Pereira, K. Pietrzak, F.-X. Standaert, and Y. Yu. "Leftover Hash Lemma, Revisited". In:*CRYPTO 2011*. Ed. by P. Rogaway. LNCS 6841. Springer, 2011, pp. 1–20. https://doi.org/10.1007/978-3-642-22792-9_1.
10. J. Bartusek, S. Garg, D. Masny, and P. Mukherjee. "Reusable two-round MPC from DDH". In: *TCC 2020*. Ed. by R. Pass and K. Pietrzak. LNCS 12551. Springer, 2020, pp. 320–348. https://doi.org/10.1007/978-3-030-64378-2_12.
11. D. Beaver. "Correlated pseudorandomness and the complexity of private computations". In: *STOC 1996*. Ed. by G. L. Miller. ACM, 1996, pp. 479–488. https://doi.org/10.1145/237814.237996.
12. D. Beaver. "Precomputing oblivious transfer". In: *CRYPTO 1995*. Ed. by D. Coppersmith. LNCS 963. Springer, 1995, pp. 97–109. https://doi.org/10.1007/3-540-44750-4_8.
13. M. Bellare and P. Rogaway. "Random oracles are practical: A paradigm for designing efficient protocols". In: *CCS 1993*. 1993, pp. 62–73. https://doi.org/10.1201/9781420010756.
14. E. Biham. "New types of cryptanalytic attacks using related keys". In: *EUROCRYPT 1993*. Ed. by T. Helleseth. LNCS 765. Springer, 1994, pp. 398–409. https://doi.org/10.1007/3-540-48285-7_34.

15. D. Boneh, Y. Ishai, A. Passelègue, A. Sahai, and D. J. Wu. "Exploring crypto dark matter: New simple PRF candidates and their applications". In: *TCC 2018*. Ed. by A. Beimel and S. Dziembowski. LNCS 11240.Springer, 2018, pp. 699–729. https://doi.org/10.1007/978-3-030-03810-6_25
16. D. Boneh and B. Waters. "Constrained pseudorandom functions and their applications". In:*ASIACRYPT 2013*. Ed. by K. Sako and P. Sarkar. LNCS 8270. Springer, 2013, pp. 280–300. https://doi.org/10.1007/978-3-642-42045-0_15.
17. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, N. Resch, and P. Scholl. "Correlated pseudorandomness from expand-accumulate codes".In: *CRYPTO 2022*. Ed. by Y. Dodis and T. Shrimpton. LNCS 13508.Springer, 2022, pp. 603–633. https://doi.org/10.1007/978-3-031-15979-4_21.
18. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl. "Efficient two-round OT extension and silent non-interactive secure computation". In: *CCS 2019*. Ed. by L. Cavallaro, J. Kinder, X.Wang, and J. Katz. ACM, 2019, pp. 291–308. https://doi.org/10.1145/3319535.3354255.
19. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. "Correlated pseudorandom functions from variable-density LPN". In:*FOCS 2020*. Ed. by S. Irani. IEEE, 2020, pp. 1069–1080. https://doi.org/10.1109/FOCS46700.2020.00103.
20. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. "Efficient pseudorandom correlation generators: Silent OT extension and more". In: *CRYPTO 2019*. Ed. by A. Boldyreva and D. Micciancio. LNCS 11694. Springer, 2019, pp. 489–518. https://doi.org/10.1007/978-3-030-26954-8_16.
21. E. Boyle, S. Goldwasser, and I. Ivan."Functional signatures and pseudorandom functions". In: *PKC 2014*. Ed. by H. Krawczyk. LNCS 8383. Springer, 2014, pp. 501–519. https://doi.org/10.1007/978-3-642-54631-0_29
22. E. Boyle, L. Kohl, and P. Scholl. "Homomorphic secret sharing from lattices without FHE". In:*EUROCRYPT 2019*. Ed. by Y. Ishai and V. Rijmen. LNCS 11477. Springer, 2019, pp. 3–33. https://doi.org/10.1007/978-3-030-17656-3_1.
23. Z. Brakerski, R. Tsabary, V. Vaikuntanathan, and H. Wee. "Private constrained PRFs (and more) from LWE". In: *TCC 2017*. Ed. by Y. Kalai and L. Reyzin. LNCS 10677. Springer, 2017, pp. 264–302. https://doi.org/10.1007/978-3-319-70500-2_10.
24. Z. Brakerski and V. Vaikuntanathan. "Constrained Key-Homomorphic PRFs from Standard Lattice Assumptions: Or: How to Secretly Embed a Circuit in Your PRF". In: *TCC 2015*. Ed. by Y. Dodis and J. B. Nielsen. LNCS 9015. Springer, 2015, pp. 1–30. https://doi.org/10.1007/978-3-662-46497-7_1.
25. D. Bui, G. Couteau, P. Meyer, A. Passel'egue, and M. Riahinia. "Fast Public-Key Silent OT and More from Constrained Naor-Reingold". In: *EUROCRYPT 2024*. Ed. by M. Joye and G. Leander. LNCS 14656. Springer, 2024, pp. 88–118. https://doi.org/10.1007/978-3-031-58751-1_4.
26. R. Canetti and Y. Chen. "Constraint-hiding constrained PRFs for NC^1 from LWE". In: *EUROCRYPT 2017*. Ed. by J. Coron and J. B. Nielsen. LNCS 10210. Springer, 2017, pp. 446–476. https://doi.org/10.1007/978-3-319-56620-7_16.
27. Y. Chen, V. Vaikuntanathan, and H. Wee. "GGH15 beyond permutation branching programs: proofs, attacks, and candidates". In: *CRYPTO 2018*. Ed. by H. Shacham and A. Boldyreva. LNCS 10992. Springer, 2018, pp. 577–607. https://doi.org/10.1007/978-3-319-96881-0_20.
28. M. Ciampi, R. Ostrovsky, L. Siniscalchi, and H. Waldner. "List oblivious transfer and applications to round-optimal black-box multiparty coin tossing". In: *CRYPTO 2023*. Ed. by H. Handschuh and A. Lysyanskaya. LNCS 14081. Springer, 2023, pp. 459–488. https://doi.org/10.1007/978-3-031-38557-5_15.

29. G. Couteau, L. Devadas, S. Devadas, A. Koch, and S. Servan-Schreiber. *QuietOT: Lightweight Oblivious Transfer with a Public-Key Setup*. Full version. 2024. Cryptology ePrint Archive, Report 2024/1079.
30. G. Couteau and C. Ducros. "Pseudorandom Correlation Functions from Variable-Density LPN, Revisited". In: *PKC 2023*. Ed. by A. Boldyreva and V. Kolesnikov. LNCS 13941. Springer, 2023, pp. 221–250. https://doi.org/10.1007/978-3-031-31371-4_8.
31. G. Couteau, A. Dupin, P. Méaux, M. Rossi, and Y. Rotella. "On the concrete security of Goldreich's pseudorandom generator". In: *ASIACRYPT 2018*. Ed. by T. Peyrin and S. D. Galbraith. LNCS 11273. Springer, 2018, pp. 96–124. https://doi.org/10.1007/978-3-030-03329-3_4.
32. G. Couteau, P. Meyer, A. Passel'egue, and M. Riahinia. "Constrained Pseudorandom Functions from Homomorphic Secret Sharing". In: *EUROCRYPT 2023*. Ed. by C. Hazay and M. Stam. LNCS 14006. Springer, 2023, pp. 194–224. https://doi.org/10.1007/978-3-031-30620-4_7.
33. G. Couteau, P. Rindal, and S. Raghuraman. "Silver: Silent VOLE and Oblivious Transfer from Hardness of Decoding Structured LDPC Codes". In: *CRYPTO 2021*. Ed. by T. Malkin and C. Peikert. LNCS 12827. Springer,2021, pp. 502–534. https://doi.org/10.1007/978-3-030-84252-9_17.
34. G. Couteau and M. Zarezadeh. "Non-interactive Secure Computation of Inner-Product from LPN and LWE". In: *ASIACRYPT 2022*. Ed. by S. Agrawal and D. Lin. LNCS 13791. Springer, 2022, pp. 474–503. https://doi.org/10.1007/978-3-031-22963-3_16.
35. A. Davidson, S. Katsumata, R. Nishimaki, S. Yamada, and T. Yamakawa. "Adaptively secure constrained pseudorandom functions in the standard model". In: *CRYPTO 2020*. Ed. by D. Micciancio and T. Ristenpart. LNCS 12170. Springer, 2020, pp. 559–589. https://doi.org/10.1007/978-3-030-56784-2_19.
36. L. de Castro, C. Juvekar, and V. Vaikuntanathan. "Fast vector oblivious linear evaluation from ring learning with errors". In: *WAHC 2021: Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. 2021, pp. 29–41. https://doi.org/10.1145/3474366.3486928
37. W. Diffie and M. Hellman. "New directions in cryptography". In:*IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. https://doi.org/10.1109/TIT.1976.1055638.
38. Y. Dodis, S. Halevi, R. D. Rothblum, and D. Wichs. "Spooky encryption and its applications". In: *CRYPTO 2016*. Ed. by M. Robshaw and J. Katz. LNCS 9816. Springer, 2016, pp. 93–122. https://doi.org/10.1007/978-3-662-53015-3_4.
39. P. Elias. "Error-correcting codes for list decoding". In: *IEEE Transactions on Information Theory* 37.1 (1991), pp. 5–12. https://doi.org/10.1109/18.61123.
40. S. Garg, M. Mahmoody, D. Masny, and I. Meckler. "On the round complexity of OT extension". In: *CRYPTO 2018*. Ed. by H. Shacham and A. Boldyreva. LNCS 10993. Springer, 2018, pp. 545–574. https://doi.org/10.1007/978-3-319-96878-0_19.
41. O. Goldreich. "Candidate one-way functions based on expander graphs". In:*Studies in Complexity and Cryptography*. Ed. by O. Goldreich. LNCS 6650. Springer, 2011, pp. 76–87. https://doi.org/10.1007/978-3-642-22670-0_10.
42. O. Goldreich, S. Micali, and A. Wigderson. "How to play any mental game, or a completeness theorem for protocols with honest majority". In: *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*. Ed. by O. Goldreich. ACM, 2019, pp. 307–328.https://doi.org/10.1145/3335741.3335755.

43. C. Guo, J. Katz, X. Wang, and Y. Yu. “Efficient and Secure Multiparty Computation from Fixed-Key Block Ciphers”. In: *SP 2020*. IEEE, 2020, pp. 825–841. https://doi.org/10.1109/SP40000.2020.00016.
44. J. HÅstad, R. Impagliazzo, L. A. Levin, and M. Luby. “A Pseudorandom Generator from any One-way Function”. In: *SIAM Journal on Computing* 28.4 (1999), pp. 1364–1396. https://doi.org/10.1137/S0097539793244708.
45. R. Impagliazzo and S. Rudich. “Limits on the provable consequences of one-way permutations”. In:*STOC 1989*. Ed. by D. S. Johnson. ACM, 1989, pp. 44–61. https://doi.org/10.1145/73007.73012.
46. J. Indigo and D. Smart. *Page Weight: 2022:* The Web Almanac by HTTP Archive. 2022. url: https://almanac.httparchive.org/en/2022/page-weight (visited on 02/29/2024).
47. Y. Ishai, J. Kilian, K. Nissim, and E. Petrank. “Extending oblivious transfers efficiently”. In: *CRYPTO 2003*. Ed. by D. Boneh. LNCS 2729. Springer, 2003, pp. 145–161. https://doi.org/10.1007/978-3-540-45146-4_9.
48. book J. Katz and Y. Lindell. *Introduction to modern cryptography: principles and protocols*. 1st ed. Chapman and Hall/CRC, 2007. https://doi.org/10.1201/9781420010756.
49. . A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. “Delegatable pseudorandom functions and applications”. In: *CCS 2013*. Ed. by A. Sadeghi, V. D. Gligor, and M. Yung. ACM, 2013, pp. 669–684. https://doi.org/10.1145/2508859.2516668.
50. J. Kilian. “Founding cryptography on oblivious transfer”. In: *STOC 1988*. Ed. by J. Simon. ACM, 1988, pp. 20–31. https://doi.org/10.1145/62212.62215.
51. V. Lyubashevsky, C. Peikert, and O. Regev. “A toolkit for ring-LWE cryptography”. In: *EUROCRYPT 2013*. Ed. by T. Johansson and P. Q. Nguyen. LNCS 7881. Springer, 2013, pp. 35–54. https://doi.org/10.1007/978-3-642-38348-9_3.
52. V. Lyubashevsky, C. Peikert, and O. Regev. “On ideal lattices and learning with errors over rings”. In: *EUROCRYPT 2010*. Ed. by H. Gilbert. LNCS 6110. Springer, 2010, pp. 1–23. https://doi.org/10.1007/978-3-642-13190-5_1.
53. S. J. Menon and D. J. Wu. “SPIRAL: Fast, high-rate single-server PIR via FHE composition”. In: *SP 2022*. IEEE, 2022, pp. 930–947. https://doi.org/10.1109/SP46214.2022.9833700.
54. M. Naor and O. Reingold. “Number-theoretic constructions of efficient pseudorandom functions”. In: *Journal of the ACM* 51.2 (2004), pp. 231–262. https://doi.org/10.1145/972639.972643.
55. C. Orlandi, P. Scholl, and S. Yakoubov. “The rise of Paillier: homomorphic secret sharing and public-key silent OT”. In:*EUROCRYPT 2021*. Ed. by A. Canteaut and F. Standaert. LNCS 12696. Springer, 2021, pp. 678–708. https://doi.org/10.1007/978-3-030-77870-5_24.
56. C. Peikert and S. Shiehian. “Privately constraining and programming PRFs, the LWE way”. In: *PKC 2018*. Ed. by M. Abdalla and R. Dahab. LNCS 10770. Springer, 2018, pp. 675–701. https://doi.org/10.1007/978-3-319-76581-5_23.
57. S. Raghuraman, P. Rindal, and T. Tanguy. “Expand-Convolute Codes for Pseudorandom Correlation Generators from LPN”. In: *CRYPTO 2023*. Ed. by H. Handschuh and A. Lysyanskaya. LNCS 14084. Springer, 2023,pp. 602–632. https://doi.org/10.1007/978-3-031-38551-3_19.
58. P. Rindal and L. Roy. *libOTe: an efficient, portable, and easy to use Oblivious Transfer Library*. url: https://github.com/osu-crypto/libOTe (visited on 01/31/2024).

59. L. Roy. "SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the Minicrypt model". In: *CRYPTO 2022*. Ed. by Y. Dodis and T. Shrimpton. LNCS 12507. Springer, 2022, pp. 657–687. https://doi.org/10.1007/978-3-031-15802-5_23.
60. P. Schoppmann, A. Gascón, L. Reichert, and M. Raykova. "Distributed Vector-OLE: Improved constructions and implementation". In: *CCS 2019*.Ed. by L. Cavallaro, J. Kinder, X.Wang, and J. Katz. ACM, 2019, pp. 1055–1072. https://doi.org/10.1145/3319535.3363228.
61. S. Servan-Schreiber. *Constrained Pseudorandom Functions for Inner-Product Predicates from Weaker Assumptions*. 2024. Cryptology ePrint Archive, Report 2024/058.
62. The OpenSSL Project.*OpenSSL: Cryptography and SSL/TLS Toolkit*. 2024. url: https://www.openssl.org/ (visited on 02/12/2024).
63. A. Ünal. *New Baselines for Local Pseudorandom Number Generators by Field Extensions*. 2023. Cryptology ePrint Archive, Report 2023/550.
64. J. Yang, Q. Guo, T. Johansson, and M. Lentmaier. "Revisiting the concrete security of Goldreich's pseudorandom generator". In:*IEEE Transactions on Information Theory* 68.2 (2021), pp. 1329–1354. https://doi.org/10.1109/TIT.2021.3128315.
65. K. Yang, C.Weng, X. Lan, J. Zhang, and X. Wang. "Ferret: Fast extension for correlated OT with small communication". In: *CCS 2020*. Ed. by J. Ligatti, X. Ou, J. Katz, and G. Vigna. ACM, 2020, pp. 1607–1626. https://doi.org/10.1145/3372297.3417276.

Constrained Pseudorandom Functions for Inner-Product Predicates from Weaker Assumptions

Sacha Servan-Schreiber(✉)

MIT, Cambridge, MA, USA
3s@mit.edu

Abstract. In this paper, we provide a novel framework for constructing Constrained Pseudorandom Functions (CPRFs) with inner-product constraint predicates, using ideas from subtractive secret sharing and related-key-attack security.

Our framework can be instantiated using a random oracle or any suitable Related-Key-Attack (RKA) secure pseudorandom function. This results in three new CPRF constructions:

1. an adaptively-secure construction in the random oracle model;
2. a selectively-secure construction under the DDH assumption; and
3. a selectively-secure construction with a polynomial domain under the assumption that one-way functions exist.

All three instantiations are constraint-hiding and support inner-product predicates, leading to the first constructions of such expressive CPRFs under each corresponding assumption. Moreover, while the OWF-based construction is primarily of theoretical interest, the random oracle and DDH-based constructions are concretely efficient.

1 Introduction

Constrained pseudorandom functions (CPRFs) [11,17,48] are pseudorandom functions (PRFs) with a "default mode" associated with a master key msk, and a "constrained mode" associated with a constrained key csk defined over a predicate C. The constrained key csk can be used to compute the same "default mode" value of the PRF for all inputs x where $C(x) = 0$. However, for all inputs x where $C(x) \neq 0$, the constrained key csk can only be used to compute a pseudorandom value that is computationally independent of the PRF value under msk.

In the basic definition of CPRFs, the constrained key csk can reveal the predicate C (i.e., all inputs x where $C(x) = 0$). For example, the GGM PRF [42], admits *puncturing* constraints [11,17,48], where the constraint C is a point function that outputs 0 on all-but-one input. In the GGM PRF, csk reveals the punctured point to the constraint key holder. An enhanced definition of CPRFs, first

This work was done in part while at Microsoft Research, New England.

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15485, pp. 232–265, 2025.
https://doi.org/10.1007/978-981-96-0888-1_8

formalized by Boneh, Lewi, and Wu [15] (PKC 2017), requires csk to hide C, and is much more challenging to achieve, even for simple constraints [15,28,36].

Constructing CPRFs for expressive constraint classes under standard assumptions has proven to be a challenging task. Several constructions exist for simple constraint classes, such as prefix-matching, bit-fixing, and constraints expressible by t-CNF formulas (with constant t) under various assumptions, including the minimal assumption that one-way functions exist (see the excellent survey of related works in [36, Appendix A]). However, even slightly more expressive constraints, such as constraints represented by inner products, constant-degree polynomials, or circuits in NC^1 (the class of functions computable by logarithmic-depth circuits), appear to be much more challenging to construct from standard assumptions [3,28,30,32].

In a recent work, Couteau, Meyer, Passelègue, and Riahinia [32] (Eurocrypt 2023) were able to realize CPRFs for NC^1 from DCR (but without the constraint-hiding property), as well as *constraint-hiding* CPRF with inner-product constraint predicates, through an elegant connection to homomorphic secret sharing [19,20,22,54]. In contrast, *constraint-hiding* CPRFs for NC^1 are only known under LWE [28,30,55] (or indistinguishability obfuscation [15,29]) and can even *imply* indistinguishability obfuscation in certain cases [28]. Therefore, the result of Couteau et al. significantly pushes the constraint expressivity of CPRFs under the Decisional Composite Residuosity (DCR) assumption. Prior to their result, the only known constructions for constraint-hiding CPRFs with sufficiently powerful constraint predicates to evaluate inner-product constraints required either the learning with errors (LWE) assumption or non-standard assumptions [15,28,55]. However, in contrast to other constraint predicates that can be realized from one-way functions [11,17,36,48], there is still a significant gap in our understanding of which assumptions are necessary for realizing CPRFs for more expressive constraint classes, such as inner-product and NC^1 predicates.

Motivation. In this paper, we revisit the assumptions required to construct constraint-hiding CPRFs for inner-product constraint classes. This is motivated by the existence of CPRFs for NC^1 from Diffie-Hellman-style assumptions [3], as well as constraint-hiding CPRFs for bit-fixing and (constant sized) t-CNF formulas from the minimal assumption that one-way functions exist [36]. Understanding what assumptions are required to realize sufficiently expressive CPRFs can shed light on realizing closely related "high-end" cryptographic primitives such as functional encryption [28,39], searchable symmetric encryption [15], attribute-based encryption [3], and even obfuscation [28]. Specifically, in this paper, we ask:

Under what assumptions do constrained PRFs with inner-product predicates exist?

The motivation for studying inner-product constraints is that they can be used to construct CPRFs with constraint predicates represented by constant-degree polynomials and extensions thereof (see the full version of this work [60]), and are of interest both as a theoretical object and as a practical tool.

From a theoretical lens, the fact that inner-product predicates lie somewhere in between t-CNF and NC^1 predicates in terms of expressivity, motivates the study of CPRFs for inner-product predicates under weaker assumptions, with the goal of potentially finding new techniques that could lead to more expressive constraints under weaker assumptions. This was also the motivation behind Attrapadung et al. [3] and other works examining the assumptions required to build CPRFs. Indeed, Davidson et al. [36] prove that CPRFs for inner-product predicates imply CPRFs for constant t-CNFs predicates (see [36, Appendix C]), which in turn imply CPRFs for bit-fixing predicates.

From a practical perspective, the current lack of any *concretely efficient* CPRF constructions for inner-product predicates,[1] motivates the quest of finding assumptions under which efficient constructions can be realized. This is especially motivated by the hope that concretely efficient constructions of CPRFs for inner-product predicates will lead to interesting real-world applications, as has been the case for the concretely efficient constructions of CPRFs admitting puncturing constraints (e.g., [5,6,16,23,38,44,50,52,58,59,61]).

Contributions. In this paper, we make the following three contributions:

New Constructions from New Assumptions. We construct the first CPRFs for inner-product predicates with (1) adaptive security in the random oracle model, (2) selective security under the Decisional Diffie-Hellman (DDH) assumption, and (3) selective security with a polynomial input domain under the minimal assumption that One-way Functions (OWFs) exist. All three of our results push the frontier of what was previously known theoretically on CPRFs. Moreover, our constructions are all constraint-hiding by default.

A Simple Framework. We provide a simple framework that exploits the properties of subtractive secret sharing to construct CPRFs for inner-product predicates. Our framework makes explicit several ideas that have been used implicitly in many prior works on CPRFs (e.g., [3,25,26,55]), and may prove useful in obtaining more results in the future.

An Implementation. Due to the simplicity of our building blocks, we show that our constructions result in the first *practical* constraint-hiding CPRFs under standard assumptions. We implement and benchmark our constructions, proving that they are concretely efficient. (All prior constructions of CPRFs for inner-product predicates, including the DCR-based construction of Couteau et al., require computationally-expensive machinery, making them impractical.)

Extensions and Applications. Our framework has the following applications and extensions.

1. *More complex predicates.* From inner-product constraints, we can build CPRFs for more complex predicates via generic transformations, including

[1] To the best of our knowledge, no constraint-hiding CPRF constructions have been implemented to date.

constraints represented by *constant degree polynomials* and CPRFs for the "AND" of d distinct inner-product predicates. In particular, the latter allows us to construct matrix-product constraint predicates, where the constraint is satisfied if and only if $\mathbf{Ax} = \mathbf{0}$, for a constraint matrix $\mathbf{A}$.

2. *Lower-bounds in learning theory.* In learning theory, Membership Query (MQ) learning provides a model for quantifying the "learnability" or complexity of a certain class of functions [62]. Informally, in the MQ learning framework, a learner gets oracle access to a function and must approximate the function after making a sufficient number of queries. Cohen, Goldwasser, and Vaikuntanathan [31] introduce a model they call MQ *with Restriction Access* ($\mathrm{MQ}_{\mathsf{RA}}$), where in addition to black-box membership queries, the learner obtains non-black-box access to a restricted subset of the function. Obtaining (negative) results on the learnability of a particular class in the $\mathrm{MQ}_{\mathsf{RA}}$ model can be done using a connection to constrained PRFs.

We discuss these applications further in the full version [60].

Followup Work. In a followup work, Couteau, Devadas, Devadas, Koch, and Servan-Schreiber [33] extend our CPRF construction to realize a *shiftable* CPRF, which allows the master key holder to emulate the PRF evaluation on the constrained key for different potential constraints. They then show how to use this extended CPRF to realize an efficient OT extension protocol with precomputability and a non-interactive public-key setup, which heavily exploits the concrete efficiency of our CPRF construction.

1.1 Related Work

In Table 1, we summarize known constructions of CPRFs for inner-product predicates (including existing constructions for more general predicates such as NC^1 and $\mathsf{P/poly}$) and highlight our results.

CPRFs for Inner-Product Predicates. Attrapadung et al. [3] construct constrained PRFs for NC^1 (which includes inner-product predicates) from the L-decisional Diffie-Hellman inversion (L-DDHI) in combination with DDH over the quadratic residue subgroup QR_p (they can make their construction adaptively-secure by using a random oracle instead of DDH in QR_p), but their construction is not constraint-hiding. Similarly, Couteau et al. [32] also show how to construct CPRFs for NC^1 predicates from the DCR assumption through homomorphic secret sharing (but also fail to achieve constraint privacy). Couteau et al. [32] additionally show that their techniques can be used to construct a CPRF from DDH *with a polynomially-bounded input domain*. CPRFs for more general predicates are known from multi-linear maps [11,14], indistinguishability obfuscation [4,12,15,36,45,46], and LWE [25,26,28,30,55], and can be used to instantiate CPRFs with inner-product constraints under those assumptions.

Constraint-Hiding CPRFs for Inner-Product Predicates. Davidson et al. [36] (Crypto 2020) construct (weakly) constraint hiding CPRFs for inner-product predicates from the LWE assumption. Specifically, their construction

Table 1. Related work on CPRFs for Inner-Product (IP) predicates from standard assumptions. Additional details are provided in the full version [60].

	Assumption	Security	Hiding	Predicate	Practical	Comments
[25,26,28,30,55]	LWE	Selective	✓/ ✗	$\supseteq \mathsf{NC}^1$	✗	[25] is not constraint hiding
AMNYY18 [3]	L-DDHI	Selective	✗	NC^1	✗	L-DDHI in QR_p $\wedge$ DDH in $\mathbb{G}$
AMNYY18 [3]	L-DDHI	Adaptive	✗	NC^1	✗	L-DDHI in QR_p $\wedge$ ROM
DKNYY20 [36]	LWE	Adaptive	✗	IP	✗	Is *weakly* constraint hiding
CMPR23 [32]	DCR	Selective	✓	IP	✗	
CMPR23 [32]	DDH	Selective	✓	IP	✗	Polynomial input domain
*vs.*pace-0.2cm						
Theorem 1	ROM	Adaptive	✓	IP	✓	
Theorem 3	DDH	Selective	✓	IP	✓	
Theorem 5	VDLPN	Selective	✓	IP	✗	Only for *weak* CPRFs
Theorem 8	OWF	Selective	✓	IP	✗	Polynomial input domain

satisfies a weaker privacy definition, in which the adversary does not get access to an evaluation oracle. Constraint-hiding CPRFs for more general predicates (that include inner-product predicates) are known from the LWE assumption [26, 28, 30, 55] and indistinguishability obfuscation [15]. To the best of our knowledge, Couteau et al. [32] are the first realize constraint-hiding CPRFs for inner-product predicates from a non-lattice assumption, specifically from DCR.

One-One CPRFs. Our framework (as well as some prior constructions of CPRFs [3,32,36]) shares some conceptual similarities to the construction of one-one constrained PRFs [56]—an information-theoretic primitive that can be viewed as a CPRF in the "no-evaluation security" model [3], with applications to conditional disclosure of secrets. However, their constructions cannot be used to realize the standard notion of CPRFs from standard assumptions.

1.2 Organization

In Sect. 2, we provide a technical overview highlighting the main ideas behind our framework and constructions. In Sect. 3, we cover the necessary preliminaries on CPRFs and RKA-secure PRFs. In Sect. 4, we present our framework and provide an adaptively secure CPRF construction for inner-product predicates in the random oracle model. In Sect. 5, we show that we can instantiate our framework from RKA-secure PRFs, without the need for a random oracle. In Sect. 6, we show how to instantiate our framework from one-way functions. In Sect. 7, we discuss the practical efficiency of our constructions.

2 Technical Overview

In this section, we provide an overview of our framework and constructions.

Background on CPRFs. Following prior works [26,32], for PRF domain $\mathcal{X}$ and a constraint $C\colon \mathcal{X} \to \{0,1\}$, we write $C(x) = 0$ for "true" (authorized), and $C(x) \neq 0$ for "false" (unauthorized). CPRFs consist of a master secret key msk, which can be used to evaluate the PRF on *all* inputs in the domain. From msk, it must then be possible to efficiently sample a constrained key csk for a given constraint C, which can be used to evaluate the PRF on all inputs x in the domain where $C(x) = 0$. Constraint hiding CPRFs have the added property that C remains hidden given csk. See Sect. 3 for formal definitions.

2.1 Our Approach

We now explain the main technical ideas that underpin our framework for constructing CPRFs for inner-product predicates. We start by explaining how we can use the idea of subtractive secret sharing to construct a constraint predicate C for inner-product predicates, inspired by Couteau et al.

The Power of Subtractive Secret Sharing. Subtractive secret shares of a value s, which we denote by s_0 and s_1, have the property that $s_0 - s_1 = s$ (over $\mathbb{Z}$). By splitting s into two random shares s_0 and s_1, individually each share is independent of the secret s. To use subtractive secret sharing to construct CPRFs, the main idea is to exploit the *symmetry* between the two shares. Specifically, consider what happens when the secret s is zero. Because we have that $s_0 - s_1 = 0$, it follows that $s_0 = s_1$. This symmetry present in subtractive secret shares has enabled many efficient techniques for distributed computations [18–22,24,40,54], and surprisingly, also applies to CPRFs [32]. Specifically, consider the inner-product constraint $C_{\mathbf{z}}$ parameterized by a vector $\mathbf{z}$ and defined as $C_{\mathbf{z}}(\mathbf{x}) = \langle \mathbf{z}, \mathbf{x} \rangle$. Next, denote subtractive secret shares of the constraint vector $\mathbf{z}$ by $\mathbf{z_0}$ and $\mathbf{z_1}$, such that $\mathbf{z_0} - \mathbf{z_1} = \mathbf{z}$. Thanks to the aforementioned symmetry property, for all input vectors $\mathbf{x}$:

- If $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ (i.e., $C_{\mathbf{z}}(\mathbf{x}) = 0$, authorized), then $\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$, and
- If $\langle \mathbf{z}, \mathbf{x} \rangle \neq 0$ (i.e., $C_{\mathbf{z}}(\mathbf{x}) \neq 0$, unauthorized), then $\langle \mathbf{z_0}, \mathbf{x} \rangle \neq \langle \mathbf{z_1}, \mathbf{x} \rangle$.

In words, the constraint is satisfied if and only if both shares of the inner product are equal. Moreover, note that $\mathbf{z_1}$ can be sampled *after* $\mathbf{z_0}$, because $\mathbf{z_0}$ is a random value independent of the "secret" constraint $\mathbf{z}$. We now describe how we can use these properties of subtractive secret sharing to construct a CPRF.

Initial Attempt (Not Secure). Our first idea, which unfortunately turns out to be not secure, is to let the master secret key $\mathsf{msk} = \mathbf{z_0}$, for a random $\mathbf{z_0}$. Then, for a given constraint vector $\mathbf{z}$, the constrained key is computed (on-the-fly) as $\mathsf{csk} = \mathbf{z_1}$, where $\mathbf{z_1} = \mathbf{z_0} - \mathbf{z}$. The intuition is that for all $\mathbf{x}$ where $\langle \mathbf{z}, \mathbf{x} \rangle = 0$ (i.e., for all authorized $\mathbf{x}$), both the master secret key and the constrained key can be used to derive *the same* key k. Specifically, we can simply let $k = \langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$. Using the key k, in conjunction with any PRF F, we can define the output of the evaluation on the input $\mathbf{x}$ to be $F_k(\mathbf{x})$. Additionally, for all $\mathbf{x}$ where $\langle \mathbf{z}, \mathbf{x} \rangle \neq 0$ (i.e., for all unauthorized $\mathbf{x}$), the master key and constrained key derive *different* PRF keys, which results in the constrained key outputting a pseudorandom value.

Unfortunately, while this initial attempt provides the necessary correctness properties, it is not secure for the following two reasons:

1. the CPRF adversary, knowing the constraint $\mathbf{z}$ and given $\mathbf{z_1}$ can trivially recover $\mathbf{z_0}$ (the master secret key) simply by computing $\mathbf{z_0} = \mathbf{z_1} + \mathbf{z}$, and
2. in the case where $\langle \mathbf{z}, \mathbf{x} \rangle \neq 0$, the derived key is still *related* to the master key msk, in that $\langle \mathbf{z_1}, \mathbf{x} \rangle = \langle \mathbf{z_0}, \mathbf{x} \rangle - \langle \mathbf{z}, \mathbf{x} \rangle$.

Couteau et al. [32] resolves these two issues by resorting to HSS. In particular, they only use the value of $\langle \mathbf{z}, \mathbf{x} \rangle$ (which each party can compute a share of given $\mathbf{z_0}$ and $\mathbf{z_1}$, respectively) as a conditional mask in a HSS computation that computes a PRF. As such, they require evaluating a PRF inside of HSS which makes their construction impractical. This is where our approach diverges from the one of Couteau et al., which we explain next.

Second Attempt (Secure). To fix our initial attempt, we must first prevent the adversary from recovering $\mathbf{z_0}$ (the master secret key) from the constrained key $\mathbf{z_1}$, while still guaranteeing the necessary property that $\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle$ whenever $\langle \mathbf{z}, \mathbf{x} \rangle = 0$. To achieve this, we exploit the linearity of inner products. Specifically, let $\mathbb{F}$ be a finite field of order at least 2^λ, for a security parameter λ. As before, we let $\mathsf{msk} := \mathbf{z_0}$, for a random $\mathbf{z_0} \in \mathbb{F}^\ell$. However, now we let $\mathsf{csk} := \mathbf{z_1}$, where $\mathbf{z_1} := \mathbf{z_0} - \Delta\mathbf{z}$, for a random scalar "shift" $\Delta \in \mathbb{F}$. Notice that when $\langle \mathbf{z}, \mathbf{x} \rangle = 0$,

$$\langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_0}, \mathbf{x} \rangle - \Delta \langle \mathbf{z}, \mathbf{x} \rangle = \langle \mathbf{z_0}, \mathbf{x} \rangle - \underbrace{\langle \Delta\mathbf{z}, \mathbf{x} \rangle}_{\substack{\text{By linearity of} \\ \text{inner products}}} = \langle \mathbf{z_1}, \mathbf{x} \rangle ,$$

which still guarantees that the master secret key and constrained key can be used to derive the same PRF key k, whenever $C(\mathbf{x}) = 0$. Moreover, because Δ is uniformly random over $\mathbb{F}$ (which has order at least 2^λ), $\mathbf{z_1}$ cannot be used to recover $\mathbf{z_0}$, even with knowledge of the constraint $\mathbf{z}$, thereby preventing the CPRF adversary from recovering the master secret key msk from the constrained key csk.

Now, with the random shift Δ, we ensure that the constrained key csk does not leak the master secret key, and forms the basis for our framework described in Sect. 4. However, we are still left with the second problem we identified in our initial attempt: The derived PRF keys are still *related* to the master secret key, which does *not* guarantee that the resulting PRF evaluation is pseudorandom to the adversary. To deal with this, we can use the random oracle model.

Construction in the Random Oracle Model. One simple way to instantiate the CPRF with correlated keys is to instantiate the PRF with a random oracle H. This forms the basis for our first instantiation, which we describe in Sect. 4.1. In a nutshell, we show that, if we use the derived key $k = \langle \mathbf{z_1}, \mathbf{x} \rangle$ with a random oracle H as the PRF, then the construction $F_k(\mathbf{x}) := H(k, \mathbf{x})$ is a secure CPRF. Specifically, the random oracle ensures that each evaluation is uniformly random, while still guaranteeing both the master secret key and the constrained key derive the same k when the constraint is satisfied.

Removing the Random Oracle with an RKA-Secure PRF. To remove the random oracle requirement, we show that we can use a "special" PRF that remains provably secure when evaluated with different *related* keys. Such PRFs are known as Related-Key-Attack (RKA) secure PRFs [8,9] and have been studied extensively [1,2,7,8,13,23,34,41,43,51], yielding several constructions to choose from. This result is rather surprising, since prior works that require notions of correlation-robustness (e.g., [47,49,57]) could only be constructed from more powerful assumptions. In contrast, we show that constructing CPRFs with inner-product constraints requires a much weaker flavor of correlation-robustness satisfied by RKA-secure PRFs with affine key-derivation functions. In particular, this weaker notion of correlation-robustness can be instantiated unconditionally leading to our one-way function based CPRF construction in Sect. 6.

Suitable RKA-Secure PRFs. As we have informally shown above, a fully "RKA-secure" PRF can be realized with a random oracle to remove correlations in the keys. However, constructions of RKA-secure PRFs exist from several standard assumptions. These constructions achieve security against adversaries that can adaptively query the PRF when keyed on arbitrary functions of the secret key. In particular, we require RKA-security against *affine* functions of the key (see Sect. 3 for definitions), which is a stronger notion compared to standard RKA-security against *additive* functions that is often considered in the literature. The affine function requirement eliminates many RKA-secure PRF constructions (e.g., [2,7,8,13,34,41,51]), leaving us only with the DDH-based RKA-secure PRF for affine functions of Abdalla et al. [1].

The DDH-based RKA-secure PRF forms the basis for our first instantiation in the standard model. However, we also show that we can use any (weak) PRF[2] that is RKA-secure against additive functions to instantiate our framework and obtain a (weak) CPRF for inner-product predicates. In particular, this allows us to use the VDLPN-based RKA-secure (weak) PRF of Boyle et al. [23].

Additionally, we show that we can adapt the one-way function based RKA-secure PRF of Applebaum and Widder [2] to instantiate our framework (under certain restrictions). Specifically, the PRF of Applebaum and Widder [2] is only secure against additive functions and requires the number of related keys that the adversary queries to be apriori bounded by some polynomial t (in the security parameter). While these restriction makes their RKA-secure PRF construction have limited applications elsewhere, we find that it is just sufficiently powerful to apply to our framework provided that we bound the magnitude of the input vectors to be polynomial in t and restrict the CPRF to a polynomially-sized domain. However, a problem is that their construction is only proven RKA-secure for *additive* functions of the key, which is not suitable to instantiate our framework. Fortunately, however, we can easily adapt their result to the case of *affine* functions, making it compatible with our framework and leading to the first Minicrypt CPRF construction for inner-product predicates.

[2] A weak PRF is secure if the adversary only queries it on random inputs.

3 Preliminaries

3.1 Notation

We let λ denote the security parameter. We let $\mathbb{F}$ denote a finite field (e.g., integers mod p), $\mathbb{Z}$ denote the set of integers, and $\mathbb{N}$ denote the set of natural numbers. We let $\mathbb{F}^{\times}$ denote the set $\mathbb{F} \setminus \{0\}$. A vector $\mathbf{v} = (v_1, \ldots, v_n)$ is denoted using bold lowercase letters. Scalar multiplication with a vector is denoted $a\mathbf{v} = (av_1, \ldots, av_n)$ and the inner product between two vectors $\mathbf{a}$ and $\mathbf{b}$ is denoted $\langle \mathbf{a}, \mathbf{b} \rangle$. We let $\mathsf{poly}(\cdot)$ denote any polynomial and $\mathsf{negl}(\cdot)$ denote a negligible function. We say an algorithm $\mathcal{A}$ is *efficient* if it runs in probabilistic polynomial time. For a finite set S, we let $x \overset{\text{R}}{\leftarrow} S$ denote a uniformly random sample from S. Assignment from a possibly randomized algorithm $\mathcal{A}$ on input x is denoted $y \leftarrow \mathcal{A}(x)$ and intialization of y to the value x is denoted as $y := x$.

3.2 Constrained Pseudorandom Functions

We start by recalling the syntax and properties of constrained pseudorandom functions (CPRFs). For simplicity, we restrict the definition to 1-key, constraint-hiding CPRFs, which is the definition satisfied by our constructions. We point to Boneh et al. [15] for a more general definition of constraint-hiding CPRFs (i.e., with polynomial-key security).

Definition 1 (Constrained Pseudorandom Functions; adapted from [15,32]). *Let $\lambda \in \mathbb{N}$ be a security parameter. A Constrained Pseudorandom Function (CPRF) with key space $\mathcal{K} = \mathcal{K}_\lambda$, domain $\mathcal{X} = \mathcal{X}_\lambda$, and range $\mathcal{Y}$, that supports constraints represented by the class of circuits $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$, where $C_\lambda \colon \mathcal{X} \to \{0,1\}$, consists of the following four algorithms.*

- $\mathsf{KeyGen}(1^\lambda) \to \mathsf{msk}$. *Takes as input a security parameter λ. Outputs a master secret key $\mathsf{msk} \in \mathcal{K}$.*
- $\mathsf{Eval}(\mathsf{msk}, x) \to y$. *Takes as input the master secret key msk and input $x \in \mathcal{X}$. Outputs $y \in \mathcal{Y}$.*
- $\mathsf{Constrain}(\mathsf{msk}, C) \to \mathsf{csk}$. *Takes as input the master secret key msk and a constraint circuit $C \in \mathcal{C}$. Outputs a constrained key csk.*
- $\mathsf{CEval}(\mathsf{csk}, x) \to y$. *Takes as input the constrained key csk and an input $x \in \mathcal{X}$. Outputs $y \in \mathcal{Y}$.*

We let any auxiliary public parameters pp be an implicit input to all algorithms. A CPRF must satisfy the following correctness and security properties.

Correctness. *For all security parameters λ, all constraints $C \in \mathcal{C}$, and all inputs $x \in \mathcal{X}$ such that $C(x) = 0$ (authorized), it holds that:*

$$\Pr\left[\mathsf{Eval}(\mathsf{msk}, x) = \mathsf{CEval}(\mathsf{csk}, x) \;\middle|\; \begin{array}{l} \mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda), \\ \mathsf{csk} \leftarrow \mathsf{Constrain}(\mathsf{msk}, C) \end{array}\right] = 1 - \mathsf{negl}(\lambda).$$

(1-key, Adaptive) Security. *A CPRF is (1-key, adaptively)-secure if for all efficient adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following security experiment $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},b}(\lambda)$ is negligible in λ. Here, b denotes the challenge bit.*

1. ***Setup:*** *On input 1^λ, the challenger runs* $\mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda)$, *initializes the set $Q := \emptyset$, and runs $\mathcal{A}(1^\lambda)$.*
2. ***Pre-challenge queries:*** *$\mathcal{A}$ adaptively sends arbitrary inputs $x \in \mathcal{X}$ to the challenger. For each x, the challenger computes $y \leftarrow \mathsf{Eval}(\mathsf{msk}, x)$, sends y to $\mathcal{A}$, and proceeds to update $Q \leftarrow Q \cup \{x\}$.*
3. ***Constrain query:*** *$\mathcal{A}$ sends* one *constraint $C \in \mathcal{C}$ to the challenger. The challenger computes* $\mathsf{csk} \leftarrow \mathsf{Constrain}(\mathsf{msk}, C)$, *and sends* csk *to $\mathcal{A}$.*
4. ***Challenge query:*** *For the single challenge query, $\mathcal{A}$ sends input $x^* \in \mathcal{X}$ as its challenge query, subject to the restriction that $x^* \notin Q$ and $C(x^*) \neq 0$. If $b = 0$, the challenger computes $y^* \leftarrow \mathsf{Eval}(\mathsf{msk}, x^*)$. Else, if $b = 1$, the challenger picks $y^* \xleftarrow{R} \mathcal{Y}$. The challenger sends y^* to $\mathcal{A}$.*
5. ***Post-challenge queries:*** *$\mathcal{A}$ continues to adaptively query the challenger on inputs $x \in \mathcal{X}$, subject to the restriction that $x \neq x^*$. For each x, the challenger computes $y \leftarrow \mathsf{Eval}(\mathsf{msk}, x)$ and sends y to $\mathcal{A}$.*
6. ***Guess:*** *$\mathcal{A}$ outputs its guess b', which is the output of the experiment.*

$\mathcal{A}$ wins if $b' = b$, and its advantage $\mathsf{Adv}^{\mathsf{sec}}_{\mathcal{A}}(\lambda)$ is defined as

$$\mathsf{Adv}^{\mathsf{sec}}_{\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A},1}(\lambda) = 1]\right|,$$

where the probability is over the randomness of $\mathcal{A}$ and KeyGen.

Definition 2 (Constraint Privacy; adapted from [15,32]**).** *A CPRF is (1-key, adaptive)-constraint-hiding if for all efficient adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following security experiment $\mathsf{Exp}^{\mathsf{priv}}_{\mathcal{A},b}(\lambda)$ is negligible in λ. Here, b denotes the challenge bit.*

1. ***Setup:*** *On input 1^λ, the challenger runs* $\mathsf{msk} \leftarrow \mathsf{KeyGen}(1^\lambda)$, *initializes the set $Q := \emptyset$, and runs $\mathcal{A}(1^\lambda)$.*
2. ***Pre-challenge queries:*** *$\mathcal{A}$ adaptively sends arbitrary input values $x \in \mathcal{X}$ to the challenger. For each x, the challenger computes $y \leftarrow \mathsf{Eval}(\mathsf{msk}, x)$, sends y to $\mathcal{A}$, and proceeds to update $Q \leftarrow Q \cup \{x\}$.*
3. ***Constrain query:*** *$\mathcal{A}$ sends a* pair *of constraints $(C_0, C_1) \in \mathcal{C}^2$ to the challenger,* subject to the restriction that *$C_0(x) = C_1(x)$, for all $x \in Q$. The challenger computes* $\mathsf{csk}^* \leftarrow \mathsf{Constrain}(\mathsf{msk}, C_b)$, *and sends* csk^* *to $\mathcal{A}$.*
4. ***Post-challenge queries:*** *$\mathcal{A}$ adaptively sends arbitrary input values $x \in \mathcal{X}$ to the challenger, subject to the restriction that $C_0(x) = C_1(x)$. For each x, the challenger computes $y \leftarrow \mathsf{Eval}(\mathsf{msk}, x)$, and sends y to $\mathcal{A}$.*
5. ***Guess:*** *$\mathcal{A}$ outputs its guess b', which is the output of the experiment.*

$\mathcal{A}$ wins if $b' = b$ and its advantage $\mathsf{Adv}^{\mathsf{priv}}_{\mathcal{A}}(\lambda)$ is defined as

$$\mathsf{Adv}^{\mathsf{priv}}_{\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{priv}}_{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{priv}}_{\mathcal{A},1}(\lambda) = 1]\right|,$$

where the probability is over the randomness of $\mathcal{A}$ and KeyGen.

Definition 3 ((1-key, selective) Security). *A CPRF as defined in Definition 1 is said to be (1-key,* selectively*)-secure if the adversary commits to the constraint C* before *querying the challenger [15]. That is, $\mathcal{A}$ sends the constraint C to the challenger before issuing any pre-challenge queries. The same applies to the constraint-privacy definition (Definition 2).*

Remark 1 (Unique evaluation queries). Without loss of generality, we can restrict the PRF adversary $\mathcal{A}$ to issuing only *unique* evaluation queries (as was also done in prior PRF formalizations [2,3]). Note that the adversary is already restricted to a unique challenge query in the above definition.

3.3 RKA-Secure PRFs

Here, we formalize the notion of related-key attack (RKA)-secure PRFs.

Remark 2 (Find-then-Guess Security). We slightly modify the standard definition of RKA-secure PRFs (e.g., [8]) to better align with the syntax of constrained PRFs. In the basic definition, the adversary does not obtain evaluation queries from what is guaranteed to be the output of the PRF F on *some* key. However, we note that this extra evaluation oracle is without loss of generality, and is only added to syntactically simplify our proofs. This definition is known as the find-then-guess PRF security game [32, Definition 10] and implies the real-or-random PRF security game, albeit with a polynomial loss in security.

Definition 4 (Φ-restricted Adversaries). *An efficient RKA-PRF adversary $\mathcal{A}$ is said to be Φ-restricted if its oracle queries have a related-key derivation function ϕ chosen arbitrarily from a set of valid key derivation functions Φ.*

Definition 5 (Related-Key-Attack Secure PRFs [8]). *Let $\lambda \in \mathbb{N}$ be a security parameter and $\ell = \ell(\lambda) \in \mathsf{poly}(\lambda)$. Let $\mathcal{F} = \{F_k : \mathcal{X}_\lambda \to \mathcal{Y}\}_{k \in \mathcal{K}_\lambda}$ be a family of functions and $\Phi \colon \mathcal{K}_\lambda \to \mathcal{K}_\lambda$ be a family of related-key derivation functions. $\mathcal{F}$ is said to be an RKA-secure PRF family if for all efficient Φ-restricted adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following security experiment $\mathsf{Exp}^{\mathsf{rka}}_{\mathcal{A},b}(\lambda)$ is negligible in λ. Here, b denotes the challenge bit.*

- ***Setup:*** *On input 1^λ, the challenger samples $k \xleftarrow{R} \mathcal{K}_\lambda$, initializes the set $Q := \emptyset$, and runs $\mathcal{A}(1^\lambda)$.*
- ***Pre-challenge queries:*** *For each query (ϕ, x), the challenger computes $y \leftarrow F_{\phi(k)}(x)$, sends y to $\mathcal{A}$, and proceeds to update $Q \leftarrow Q \cup \{(\phi, x)\}$.*
- ***Challenge query:*** *For the single challenge query (ϕ^*, x^*),* subject to the restriction that *$(\phi^*, x^*) \notin Q$, the challenger proceeds based on the bit b as follows. If $b = 0$, the challenger computes $y \leftarrow F_{\phi^*(k)}(x^*)$. If $b = 1$, the challenger samples $y \xleftarrow{R} \mathcal{Y}$. The challenger then sends y to $\mathcal{A}$.*
- ***Post-challenge queries:*** *For each query (ϕ, x),* subject to the restriction that *$(\phi, x) \neq (\phi^*, x^*)$, the challenger computes $y \leftarrow F_{\phi^*(k)}(x)$, and sends y to $\mathcal{A}$.*

– ***Guess:*** *$\mathcal{A}$ outputs its guess b', which is the output of the experiment.*

$\mathcal{A}$ wins if $b' = b$ and its advantage $\mathsf{Adv}^{\mathsf{rka}}_{\mathcal{A}}(\lambda)$ is defined as

$$\mathsf{Adv}^{\mathsf{rka}}_{\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Exp}^{\mathsf{rka}}_{\mathcal{A},0}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{rka}}_{\mathcal{A},1}(\lambda) = 1]\right|,$$

where the probability is over the randomness of $\mathcal{A}$ and choice of k.

Definition 6 (Affine Related-Key Derivation Functions [1]**).** *Let $\mathbb{F}$ be a finite field and let $n \geq 1$ be an integer, let the class Φ_{aff} (aff for affine) denote the class of functions from $\mathbb{F}^n$ to $\mathbb{F}^n$ that can be separated into n component functions consisting of degree-1 univariate polynomials. That is,*

$$\Phi_{\mathsf{aff}} := \left\{ \boldsymbol{\phi} : \mathbb{F}^n \to \mathbb{F}^n \mid \begin{array}{l} \boldsymbol{\phi} = (\phi_1, \ldots, \phi_n); \\ \forall i \in [n],\ \phi_i(k_i) = \gamma_i k_i + \delta_i,\ \gamma_i \neq 0 \end{array} \right\}.$$

Note that $\gamma_i \neq 0$ is necessary to make the derivation function non-trivial.

Remark 3. Note that Φ_{aff} captures additive and multiplicative relations, which we denote by $\Phi_{+} \subset \Phi_{\mathsf{aff}}$ and $\Phi_{\times} \subset \Phi_{\mathsf{aff}}$, respectively.

4 The Basic Framework and Construction

In Construction 1, we present our basic framework for constructing CPRFs for inner-product predicates, and present an instantiation of it in the random oracle model in Sect. 4.1. We extend this framework and use it in conjunction with RKA-secure PRFs in Sect. 5 to realize CPRFs for inner-product predicates under DDH, VDLPN, and OWFs.

Construction 1 (The basic framework)**.**
Let λ be a security parameter, $\ell \geq 1$ be an integer, and $\mathbb{F}$ be a finite field of order at least 2^λ. For a key space $\mathcal{K}$ and range $\mathcal{Y}$, a suitable choice of efficiently computable deterministic function $\mathsf{map}\colon \mathbb{F} \to \mathcal{K}$, and a PRF family $\mathcal{F} = \left\{F_k\colon \mathbb{F}^\ell \to \mathcal{Y}\right\}_{k\in\mathcal{K}}$, the CPRF algorithms are defined as:

```
KeyGen(1^λ, ℓ):
  1 : $k_0 \xleftarrow{R} \mathbb{F}$
  2 : $\mathbf{z_0} \xleftarrow{R} \mathbb{F}^\ell$
  3 : $\mathsf{msk} := (k_0, \mathbf{z_0})$

Eval(msk, x):
  1 : parse $\mathsf{msk} = (k_0, \mathbf{z_0})$
  2 : $\delta_{\mathbf{x}} := \langle \mathbf{z_0}, \mathbf{x} \rangle$
  3 : $k \leftarrow \mathsf{map}(k_0 + \delta_{\mathbf{x}})$
  4 : return $F_k(\mathbf{x})$

Constrain(msk, z):
  1 : parse $\mathsf{msk} = (k_0, \mathbf{z_0})$
  2 : $\Delta \xleftarrow{R} \mathbb{F}^\times$
  3 : $\mathbf{z_1} := \mathbf{z_0} - \Delta\mathbf{z}$
  4 : return $\mathsf{csk} := (k_0, \mathbf{z_1})$

CEval(csk, x):
  1 : parse $\mathsf{csk} = (k_0, \mathbf{z_1})$
  2 : $\delta_{\mathbf{x}} := \langle \mathbf{z_1}, \mathbf{x} \rangle$
  3 : $k \leftarrow \mathsf{map}(k_0 + \delta_{\mathbf{x}})$
  4 : return $F_k(\mathbf{x})$
```

4.1 Instantiation via a Random Oracle

The simplest instantiation of Construction 1 is to let $F_k(\mathbf{x}) := H(k, \mathbf{x})$ where $H \colon \mathcal{K} \times \mathbb{F}^\ell \to \mathcal{Y}$ is a random oracle. Doing so ensures that when $\langle \mathbf{z}, \mathbf{x} \rangle \neq 0$, the output is uniformly random and independent of the constrained key csk, which guarantees that the evaluation under msk is independent of csk. We prove the following theorem.

Theorem 1. *Let λ be a security parameter, $\ell \geq 1$ be any integer, $\mathbb{F}$ be a finite field of order at least 2^λ, and map be any entropy-preserving map. Construction 1 is a (1-key, adaptively-secure, constraint-hiding) CPRF in the random oracle model when $\mathcal{F} = \left\{F_k \colon \mathbb{F}^\ell \to \mathcal{Y}\right\}_{k \in \mathcal{K}}$ is a PRF family, where $F_k(\mathbf{x}) := H(k, \mathbf{x})$ for all $k \in \mathcal{K}$ and $\mathbf{x} \in \mathbb{F}^\ell$, and where $H \colon \mathcal{K} \times \mathbb{F}^\ell \to \mathcal{Y}$ is a random oracle.*

Proof. We prove each required property in turn.

Correctness. Correctness follows from the intuition presented in Sect. 2. For all constraints $\mathbf{z}$ and inputs $\mathbf{x}$, whenever $\langle \mathbf{z}, \mathbf{x} \rangle = 0$, we have that

$$\delta_{\mathbf{x}} = \langle \mathbf{z_0}, \mathbf{x} \rangle = \langle \mathbf{z_0}, \mathbf{x} \rangle + \langle \mathbf{z}, \mathbf{x} \rangle = \langle \mathbf{z_0}, \mathbf{x} \rangle + \langle \Delta \mathbf{z}, \mathbf{x} \rangle = \langle \mathbf{z_1}, \mathbf{x} \rangle .$$

Therefore, Eval and CEval (of Construction 1) compute the same key k, because both Eval and CEval add the same shift $\delta_{\mathbf{x}}$ to the starting key k_0. It then follows that the evaluation is identical under the master key and the constrained key given that F_k is deterministic.

(1-key, Adaptive) Security. We prove security by a reduction to the security of the (non-constrained) PRF $F_k(\cdot)$ used to instantiate Construction 1. Our proof consists of a sequence of hybrid games.

Hybrid $\mathcal{H}_0$. This hybrid consists of the (1-key, adaptive) CPRF security game. We note that here, the challenger provides an oracle $\mathcal{O}_H$ via which the adversary $\mathcal{A}$ queries the random oracle H.

Hybrid $\mathcal{H}_1$. In this hybrid game, we place the following restrictions on the adversary: (1) Each query issued by $\mathcal{A}$ to the challenger (including queries to $\mathcal{O}_H$) must be unique, and (2) *after* issuing the constraint query, the adversary is only allowed to query the pre- and post-challenge oracles on constrained inputs.

These restrictions are without loss of generality in the 1-key setting, and have been used in prior work (e.g., [3,36]). It follows that $\mathcal{A}$'s advantage in $\mathcal{H}_1$ is identical to its advantage in $\mathcal{H}_0$.

Hybrid $\mathcal{H}_2$. In this hybrid game, the challenger starts by sampling responses to the pre-challenge evaluation queries until the constraint query is issued. That is, for a bound q_0 on the number of pre-challenge queries issued by $\mathcal{A}$, the challenger samples $v_1, \ldots, v_{q_0} \overset{\text{R}}{\leftarrow} \mathcal{Y}$. Then, the challenger responds to the i-th pre-challenge query $\mathbf{x_i}$ with v_i, and programs the random oracle $\mathcal{O}_H$ to output v_i on all future queries r where it holds that $r = k_0 + \langle \mathbf{z_0}, \mathbf{x_i} \rangle$, for some $i \in [q_0]$. After the constrain query, the challenger responds to pre-challenge queries as in $\mathcal{H}_1$.

Claim. $\mathcal{A}$'s advantage in $\mathcal{H}_2$ is at most $\mathsf{negl}(\lambda)$ larger compared to $\mathcal{H}_1$.

Proof. Let q_H be the total number of random oracle queries issued by the adversary prior to the constrain query, and let $\{r_i\}_{i\in[q_0]}$ be the queries to $\mathcal{O}_H$. We define the event bad_0 as:

$$\exists(i,j) \in [q_H] \times [q_0] \text{ such that } r_i = (k_0 + \langle \mathbf{z_0}, \mathbf{x_j}\rangle, \mathbf{x_j}) \wedge H(r_i) \neq v_j.$$

The event bad_0 corresponds to the case where the adversary happens to query the random oracle $\mathcal{O}_H$ on a "bad" input r_i *prior* to the challenger programming $\mathcal{O}_H$ to output v_i, causing the response to be inconsistent with respect to $\mathcal{H}_1$. For any given pre-challenge query $\mathbf{x_j}$, issued before the constrain query, the probability that $\mathcal{A}$ issues an r_i query to $\mathcal{O}_H$ such that $r_i = (k_0 + \langle \mathbf{z_0}, \mathbf{x_j}\rangle, \mathbf{x_j})$ is equivalent to the probability of guessing k_0, which is bounded by $\frac{1}{|\mathbb{F}|}$. Hence, by a union bound, we get that

$$\Pr_{k_0 \xleftarrow{R} \mathbb{F}}[\mathsf{bad}_0] \leq \frac{q_H \cdot q_0}{|\mathbb{F}|} \leq \frac{q_H \cdot q_0}{2^\lambda} \leq \mathsf{negl}(\lambda),$$

which bounds $\mathcal{A}$'s advantage in $\mathcal{H}_1$ to a negligible function in λ. □

Hybrid $\mathcal{H}_3$. In this hybrid game, we swap the definition of the constrained key and master key. Specifically, in this game, the challenger responds to $\mathcal{A}$'s constrain query $\mathbf{z}$ by sampling $\mathbf{z_1} \xleftarrow{R} \mathbb{F}^\ell$ and sending back $\mathsf{csk} = \mathbf{z_1}$. The challenger then samples $\Delta \xleftarrow{R} \mathbb{F}$, computes $\mathbf{z_0} = \mathbf{z_1} + \Delta\mathbf{z}$, and responds to future evaluation queries using $\mathbf{z_0}$ as the master key.

Claim. $\mathcal{A}$'s advantage in $\mathcal{H}_3$ is equivalent to its advantage in $\mathcal{H}_2$.

Proof. This change is purely syntactic and therefore does not affect the distribution of the keys. In particular, note that in $\mathcal{H}_2$, all evaluation queries prior to the constrain query are sampled independently of the master key. As such, it can be sampled at the time of the constrain query. □

Hybrid $\mathcal{H}_4$. In this hybrid game, the challenger samples $w_1, \ldots, w_{q_1} \xleftarrow{R} \mathcal{Y}$ as the responses to the q_1 pre- and post-challenge evaluation queries issued following the constrain query. Then, the challenger responds to $\mathcal{A}$'s i-th evaluation query $\mathbf{x_i}$, where $\langle \mathbf{z}, \mathbf{x_i}\rangle \neq 0$ (recall the restriction in $\mathcal{H}_0$), with w_i.

Claim. $\mathcal{A}$'s advantage in $\mathcal{H}_4$ is at most $\mathsf{negl}(\lambda)$ larger compared to $\mathcal{H}_3$.

Proof. Here, we let q_H be a bound on the total number of random oracle queries issued by the adversary throughout the game and let q_1 be a bound on the number of pre- and post-challenge evaluation queries issued *after* the constrain query. We then define the event bad_1 as:

$$\exists(i,j) \in [q_H] \times [q_1] \text{ such that } r_i = (k_0 + \langle \mathbf{z_0}, \mathbf{x_j}\rangle, \mathbf{x_j}) \wedge H(r_i) \neq w_j,$$

where r_i is a query to $\mathcal{O}_H$ issued by $\mathcal{A}$ and each $\mathbf{x_j}$ is constrained by assumption. The event bad_1 corresponds to the case where the adversary happens to query $\mathcal{O}_H$ on an input corresponding to a constrained evaluation under the master key msk, causing the response to be inconsistent with respect to the distribution in $\mathcal{H}_3$. For all post-constraint evaluation queries $\mathbf{x_j}$, where $j \in [q_1]$, define $y_j = H(k_0 + \langle \mathbf{z_0}, \mathbf{x_j} \rangle, \mathbf{x_j})$, which is computed identically to a post-constraint evaluation response in hybrid $\mathcal{H}_3$. We claim that y_j is computed independently of the constrained key $\mathsf{csk} = \mathbf{z_1}$. To see this, note that we can equivalently express y_j in terms of $\mathbf{z_1}$ as $y_j = H(k_0 + \langle \mathbf{z_1}, \mathbf{x_j} \rangle + \Delta \langle \mathbf{z}, \mathbf{x_j} \rangle, \mathbf{x_j})$, where $\langle \mathbf{z}, \mathbf{x_j} \rangle \neq 0$ by assumption. Then, because Δ is uniformly random and independent of $\mathbf{z_1}$ in $\mathcal{H}_3$, each y_j is computed using a random oracle H that is "seeded" by $\Delta \langle \mathbf{z}, \mathbf{x_j} \rangle$, which makes the response independent of $\mathbf{z_1}$. Then, to compute the probability of the event bad_1, over the choice of $\Delta \in \mathbb{F}$, we can apply a union bound over all q_1 post-constraint evaluation queries issued by $\mathcal{A}$ to get

$$\Pr_{\Delta \xleftarrow{R} \mathbb{F}} [\mathsf{bad}_1] \leq \frac{q_H \cdot q_1}{|\mathbb{F}|} \leq \frac{q_H \cdot q_1}{2^\lambda} = \mathsf{negl}(\lambda),$$

which bounds the adversary's advantage in $\mathcal{H}_4$ to a negligible function in λ. □

Hybrid $\mathcal{H}_5$. In this hybrid game, the challenger samples a uniformly random key k to answer the challenge query when the challenge bit is set to $b = 0$.

Claim. $\mathcal{A}$'s advantage in $\mathcal{H}_5$ is equivalent to its advantage in $\mathcal{H}_4$.

Proof. Note that all evaluation queries in $\mathcal{H}_4$ are sampled independently of Δ. Therefore, Δ is only used by the challenger in $\mathcal{H}_4$ to respond to the challenge query, which is equivalent to sampling a uniformly random and independent key k to answer the challenge query $\mathbf{x}^*$, given that $k_0 + \langle \mathbf{z_0}, \mathbf{x}^* \rangle = k_0 + \langle \mathbf{z_1}, \mathbf{x}^* \rangle + \Delta \langle \mathbf{z}, \mathbf{x}^* \rangle$ is a uniformly random value in $\mathbb{F}$. □

Claim. If F_k is a secure PRF, then there does not exist an efficient $\mathcal{A}$ that wins game defined in $\mathcal{H}_5$ with better than negligible advantage.

Proof. Suppose, towards contradiction, that there exists an efficient $\mathcal{A}$ that wins the game defined in $\mathcal{H}_5$ with non-negligible advantage, then there exists an efficient $\mathcal{B}$ that wins the find-then-guess PRF security game with the same advantage: $\mathcal{B}$ simply plays the role of the challenger in $\mathcal{H}_5$ and forwards the challenge query from the adversary to its own PRF challenger, responding to $\mathcal{A}$ as its own challenger does. Since the only dependence on the PRF key in $\mathcal{H}_5$ is in the challenge phase, $\mathcal{B}$ wins the PRF security game with the same advantage. □

Finally, noting that $F_k(\mathbf{x}) := H(k, \mathbf{x})$ trivially satisfies the definition of a pseudorandom function when H is a random oracle and k has sufficient entropy proves (1-key, adaptive) security.

Constraint Privacy. We must prove that for all $\mathbf{z}$ and $\mathbf{z}'$ provided by the adversary $\mathcal{A}$, the constrained key, and all evaluation and challenge queries, do not reveal whether the constraint $\mathbf{z}$ or $\mathbf{z}'$ is used by the challenger.

First, we begin by noting that, even given $(\mathbf{z}, \mathbf{z}', \Delta)$, $\mathbf{z_0} + \Delta\mathbf{z}$ is distributed identically to $\mathbf{z_0} + \Delta\mathbf{z}'$ because $\mathbf{z_0}$ is uniformly random and independent of $\mathbf{z}$ and $\mathbf{z}'$. Therefore, the constrained key, absent the evaluation queries, is efficiently simulatable regardless of the constraint chosen by the challenger.

Now, we must show that this remains the case even when the adversary is given access to the evaluation and challenge oracles. Observe that we can proceed via the same sequence of hybrids used in the security proof. Note that in the game defined in Hybrid $\mathcal{H}_5$, we can view each constrained query as being answered using a uniformly random key $k_i \in \mathcal{K}$. As such, the evaluation queries on constrained inputs (including the challenge query) are independent of the constraint, which guarantees that $\mathcal{A}$ cannot distinguish between $\mathbf{z}$ and $\mathbf{z}'$ with better than negligible advantage.

This concludes the proof constraint privacy and the proof of the theorem. ■

Remark 4 (Replacing the random oracle with a correlated-input secure hash). As noted by several prior works (e.g., [37,43,47]), the random oracle model is an overkill when all that is required is a notion of "correlation-robustness." Specifically, in our case, all we require is that H removes specific types of correlations present in its inputs. With this in mind, we can replace the random oracle H with a correlated-input secure hash (CIH) function [3,37,43,47]. At a high level, a CIH is a publicly parameterized function H whose outputs "look random and independent" to a computationally-bounded adversary, even when the inputs are correlated. Specifically, we require the CIH to be secure against *affine* correlations between the inputs. The proof of security for Theorem 1 then follows the same blueprint, but instead hinges on the correlated-input security of H to ensure that the outputs are computationally indistinguishable from uniform. Unfortunately, we are not aware of an adaptively-secure CIH function construction (to the best of our knowledge, all existing constructions are in the selective-security regime). However, we note that there exist strong connections between CIH functions and RKA-PRFs, as discussed in-depth by Goyal, O'Neill, and Rao [43].

5 Generalized Framework and Constructions

In this section, we instantiate our framework via RKA-secure PRFs. In Sect. 5.1, we start by extending the basic framework from Sect. 4 to make it more amenable with RKA-secure PRF constructions. We then prove that this framework yields constraint-hiding CPRFs from any RKA-secure PRF supporting Φ_{aff} key derivation functions. In Sect. 5.2 and 5.3, we plug in the DDH-based and VDLPN-based RKA-secure PRF constructions into the framework. We defer instantiating the framework with our OWF-based RKA-secure PRF to Sect. 6, as there we must first construct a Φ_{aff}-RKA-secure PRF from OWFs.

5.1 Extended Framework

Existing constructions of RKA-secure PRFs (e.g., [1,2,7,23]) have a key that is a *vector* of n field elements. As such, we cannot directly instantiate Construction 1 because the inner products are performed in $\mathbb{F}$ but the keys live in the vector space $\mathbb{F}^n$ (or subfield thereof). We therefore provide an *extended* version of our framework in Construction 2, that can be instantiated with the parameters of existing RKA-secure PRFs. At a high level, to accommodate keys that consist of vectors of n elements, we apply Construction 1 independently n times to derive a key for each coordinate. Formally, we capture this in Construction 2.

Construction 2 (The extended framework)**.**
Let λ be a security parameter, $n, \ell \geq 1$ be integers, and $\mathbb{F}$ be a finite field. For a key space $\mathcal{K}$ and range $\mathcal{Y}$, a suitable choice of efficiently computable deterministic function $\mathsf{map}\colon \mathbb{F}^n \to \mathcal{K}$, and a PRF family $\mathcal{F} = \left\{F_k\colon \mathbb{F}^\ell \to \mathcal{Y}\right\}_{k \in \mathcal{K}}$, the CPRF algorithms are defined as:

```
$\mathsf{KeyGen}(1^\lambda, \ell)$:
1 : $\mathbf{k_0} \xleftarrow{R} \mathbb{F}^n$
2 : foreach $i \in [n]$ :
3 :     $\mathbf{z_0}_i \xleftarrow{R} \mathbb{F}^\ell$
4 : $\mathsf{msk} := (\mathbf{k_0}, \mathbf{z_0}_1, \ldots, \mathbf{z_0}_n)$
```

```
$\mathsf{Eval}(\mathsf{msk}, \mathbf{x})$:
1 : parse $\mathsf{msk} = (\mathbf{k_0}, \mathbf{z_0}_i, \ldots, \mathbf{z_0}_n)$
2 : foreach $i \in [n]$ :
3 :     $\delta_{\mathbf{x}i} := \langle \mathbf{z_0}_i, \mathbf{x} \rangle$
4 : $\boldsymbol{\delta}_\mathbf{x} := (\delta_{\mathbf{x}1}, \ldots, \delta_{\mathbf{x}n})$
5 : $k \leftarrow \mathsf{map}(\mathbf{k_0} + \boldsymbol{\delta}_\mathbf{x})$
6 : return $F_k(\mathbf{x})$
```

```
$\mathsf{Constrain}(\mathsf{msk}, \mathbf{z})$:
1 : parse $\mathsf{msk} = (\mathbf{k_0}, \mathbf{z_0}_i, \ldots, \mathbf{z_0}_n)$
2 : foreach $i \in [n]$ :
3 :     $\Delta_i \xleftarrow{R} \mathbb{F}$
4 :     $\mathbf{z_1}_i := \mathbf{z_0}_i - \Delta_i \mathbf{z}$
5 : return $\mathsf{csk} := (\mathbf{k_0}, \mathbf{z_1}_1, \ldots, \mathbf{z_1}_n)$
```

```
$\mathsf{CEval}(\mathsf{csk}, \mathbf{x})$:
1 : parse $\mathsf{csk} := (\mathbf{k_0}, \mathbf{z_1}_1, \ldots, \mathbf{z_1}_n)$
2 : foreach $i \in [n]$ :
3 :     $\delta_{\mathbf{x}i} := \langle \mathbf{z_1}_i, \mathbf{x} \rangle$
4 : $\boldsymbol{\delta}_\mathbf{x} := (\delta_{\mathbf{x}1}, \ldots, \delta_{\mathbf{x}n})$
5 : $k \leftarrow \mathsf{map}(\mathbf{k_0} + \boldsymbol{\delta}_\mathbf{x})$
6 : return $F_k(\mathbf{x})$
```

Theorem 2. *Let $\mathbb{K}$ be a subfield of $\mathbb{F}$ and let the PRF key space $\mathcal{K} = \mathbb{K}^n$. Fix* map *to be any non-trivial ring homomorphism applied component-wise. If $\mathcal{F}$ is a family of RKA-secure pseudorandom functions with respect to affine related key derivation functions Φ_{aff}, as defined in Definition 6, then Construction 2 instantiated with $\mathcal{F}$ is a (1-key, selectively-secure, constraint-hiding) CPRF.*

Proof. We prove the required properties in turn.

Correctness. For all constraints $\mathbf{z}$ and inputs $\mathbf{x}$, whenever $\langle \mathbf{z}, \mathbf{x} \rangle = 0$, we have that $\delta_{\mathbf{x}i} = \langle \mathbf{z_0}_i, \mathbf{x} \rangle = \langle \mathbf{z_0}_i, \mathbf{x} \rangle + \Delta_i \langle \mathbf{z}, \mathbf{x} \rangle = \langle \mathbf{z_0}_i, \mathbf{x} \rangle + \langle \Delta_i \mathbf{z}, \mathbf{x} \rangle = \langle \mathbf{z_1}_i, \mathbf{x} \rangle \in \mathbb{F}$. Therefore, the resulting $\boldsymbol{\delta}_\mathbf{x}$ (as computed in Eval and CEval of Construction 1) is the same. Moreover, this holds for all $i \in [n]$, and because map is a ring homomorphism to a subfield of $\mathbb{F}$, the resulting keys are also identical when $\langle \mathbf{z}, \mathbf{x} \rangle = 0$. It then follows that the PRF evaluation is identical under the master key and the constrained key, because both Eval and CEval add the same $\boldsymbol{\delta}_{\mathbf{x}}$.

(1-key, Selective) Security. We prove security by a reduction to the RKA-security of $\mathcal{F}$. Our proof consists of a sequence of hybrid games.

Hybrid $\mathcal{H}_0$. This hybrid consists of the (1-key, selective) CPRF security game.

Hybrid $\mathcal{H}_1$. In this hybrid, the challenger first samples the constrained key and *then* samples the master key. Specifically, at the start of the game, given the constraint $\mathbf{z}$ (we're in the selective security regime), the challenger first samples the constrained key $\mathsf{csk} := (\mathbf{k_0}, \mathbf{z_1}_1, \ldots, \mathbf{z_1}_n)$, where $\mathbf{k_0} \overset{\mathrm{R}}{\leftarrow} \mathbb{F}^n$ and $\mathbf{z_1}_i \overset{\mathrm{R}}{\leftarrow} \mathbb{F}^\ell$, for all $i \in [n]$. Then, the challenger computes the master secret key as $\mathsf{msk} := (\mathbf{k_0}, \mathbf{z_0}_1, \ldots, \mathbf{z_0}_n)$, where $\mathbf{z_0}_i := \mathbf{z_1}_i + \Delta_i \mathbf{z}$ and $\Delta_i \overset{\mathrm{R}}{\leftarrow} \mathbb{F}$, for all $i \in [n]$.

Claim. $\mathcal{A}$'s advantage in $\mathcal{H}_1$ is identical to $\mathcal{A}$'s advantage in $\mathcal{H}_0$.

Proof. The claim follows immediately by observing that the distribution of msk and csk in $\mathcal{H}_1$ is identical to $\mathcal{H}_0$, because the change is merely syntactic. □

Hybrid $\mathcal{H}_2$. In this hybrid game, the challenger does not sample Δ anymore. Instead, it is given access to the following stateful oracle $\mathcal{O}_{\mathsf{rka}}$:

Oracle $\mathcal{O}_{\mathsf{rka}}$

Initialize. Sample $\Delta \overset{\mathrm{R}}{\leftarrow} \mathbb{K}^n$.

Evaluation. On input a affine function $\phi \in \Phi_{\mathsf{aff}}$ and $\mathbf{x} \in \mathbb{F}^\ell$, return $F_{\phi(\Delta)}(\mathbf{x})$.

The challenger is then defined as follows.

1. **Setup:** On input $(1^\lambda, \mathbf{z})$, $\mathcal{B}$ initializes $Q := \emptyset$, samples csk according to $\mathcal{H}_1$ by sampling $\mathbf{k_0} \overset{\mathrm{R}}{\leftarrow} \mathbb{F}^n$, and $\mathbf{z_1}_i \overset{\mathrm{R}}{\leftarrow} \mathbb{F}^\ell$, for all $i \in [n]$, and runs $\mathcal{A}$ on input $\mathsf{csk} := (\mathbf{k_0}, \mathbf{z_1}_1, \ldots, \mathbf{z_1}_n)$.
2. **Pre-challenge queries:** For each query $\mathbf{x}$ issued by $\mathcal{A}$, the challenger updates $Q \leftarrow Q \cup \{\mathbf{x}\}$, then does the following to compute y:
   - Compute $a_i := \mathsf{map}(\langle \mathbf{z}, \mathbf{x} \rangle)$ and $b_i := \mathsf{map}(\mathbf{k_0}_i + \langle \mathbf{z_1}_i, \mathbf{x} \rangle)$, for all $i \in [n]$.
   - Set $\phi \colon \mathbf{u} \mapsto \mathbf{a} \circ \mathbf{u} + \mathbf{b}$ where $\mathbf{a} := (a_1, \ldots, a_n)$ and $\mathbf{b} := (b_1, \ldots, b_n)$, where $\circ$ denotes the component wise (i.e., Hadamard) product.
   - Query $\mathcal{O}_{\mathsf{rka}}$ on input $(\phi, \mathbf{x})$, and forward the response y to $\mathcal{A}$.
     $\triangleright$ Note that y is computed by $\mathcal{O}_{\mathsf{rka}}$ as $F_{\mathbf{k}'}(\mathbf{x})$ where
     $\triangleright$ $\mathbf{k}' = \mathbf{a} \circ \Delta + \mathbf{b} \in \mathbb{K}^n = \phi(\Delta)$, for $\phi \in \Phi_{\mathsf{aff}}$.

3. **Challenge:** For the single challenge query $\mathbf{x}^*$, subject to $\langle \mathbf{z}, \mathbf{x}^* \rangle \neq 0$ and $\mathbf{x}^* \notin Q$, the challenger does the following. Sample $b \in \{0, 1\}$:
   - If $b = 0$, then
     - Compute $a_i := \mathsf{map}(\langle \mathbf{z}, \mathbf{x} \rangle)$ and $b_i := \mathsf{map}(\mathbf{k_0}_i + \langle \mathbf{z_1}_i, \mathbf{x}^* \rangle)$, for all $i \in [n]$.
     - Set $\phi^* : \mathbf{u} \mapsto \mathbf{a} \circ \mathbf{u} + \mathbf{b}$ where $\mathbf{a} := (a_1, \dots, a_n)$ and $\mathbf{b} := (b_1, \dots, b_n)$, where $\circ$ denotes the component wise product.
     - Query $\mathcal{O}_{\mathsf{rka}}$ on input $(\phi^*, \mathbf{x}^*)$, and forward the response y^* to $\mathcal{A}$.
   - Else if $b = 1$, then
     - Sample $y^* \xleftarrow{R} \mathcal{Y}$ and send y^* to $\mathcal{A}$.
4. **Post-challenge queries:** Answered identically to pre-challenge queries.

Claim. $\mathcal{A}$'s advantage in $\mathcal{H}_2$ is identical to $\mathcal{A}$'s advantage in $\mathcal{H}_1$.

Proof. The difference between $\mathcal{H}_2$ and $\mathcal{H}_1$ is again purely syntactic since each output is computed identically in both games, with the only difference being that the challenger now only has access to Δ via the oracle $\mathcal{O}_{\mathsf{rka}}$ □.

Claim. If $\mathcal{F}$ is an RKA-secure PRF for affine related key derivation functions Φ_{aff}, then there does not exist an efficient $\mathcal{A}$ that wins the game defined in $\mathcal{H}_2$ with better than negligible advantage.

Proof. Suppose, towards contradiction, that there exits an efficient adversary $\mathcal{A}$ for $\mathcal{H}_2$ that wins with non-negligible advantage. Construct an efficient Φ_{aff}-restricted adversary $\mathcal{B}$ that wins the RKA security game for the PRF F_k with the same advantage. $\mathcal{B}$ simply plays the role of the challenger in $\mathcal{H}_2$, forwarding all queries to its own challenger. Note that this makes $\mathcal{B}$'s queries Φ_{aff}-restricted. Therefore, on the one hand, when $\mathcal{B}$ is given access to a truly random function at the challenge phase, its answers are distributed identically to $\mathcal{H}_2$ when the challenger samples $b = 1$. On the other hand, when $\mathcal{B}$ is given access to a an RKA-PRF oracle, $\mathcal{B}$'s answers are distributed identically to $\mathcal{H}_2$ when the challenger samples $b = 0$ and queries $\mathcal{O}_{\mathsf{rka}}$. As such, $\mathcal{B}$ has the same advantage as $\mathcal{A}$, which contradicts the RKA-security of $\mathcal{F}$. □

This concludes the proof of (1-key, selective) security.

Constraint Privacy. For constraint privacy, we must show that if $\mathcal{F}$ is an RKA-secure PRF family, then all evaluation and challenge queries remain pseudorandom, regardless of whether constraint $\mathbf{z}$ or $\mathbf{z}'$ is used by the challenger.[3]

Again, note that $\mathbf{z_0}_i + \Delta_i \mathbf{z}$ is distributed identically to $\mathbf{z_0}_i + \Delta_i \mathbf{z}'$, thereby making the constraint key, absent the evaluation queries, efficiently simulatable regardless of the constraint chosen by the challenger. Now, we must show that this remains the case even when the adversary is given access to the evaluation oracles. We prove this via the following lemma. Roughly speaking, the lemma

[3] Recall that the adversary provides two constraints $\mathbf{z}$ and $\mathbf{z}'$.

states that if the underlying PRF is RKA-secure, then distinguishing between evaluations under two different related-key derivation functions of the PRF key contradicts the RKA security of the PRF.

Lemma 1. *Let λ be a security parameter and $\mathcal{F} = \{F_k \colon \mathcal{X} \to \mathcal{Y}\}_{k \in \mathcal{K}}$ be an RKA-secure PRF. Then, for all efficient Φ-restricted adversaries $\mathcal{A}$, the advantage in the following game is negligible in λ.*

- **Setup:** On input 1^λ, the challenger samples $k \xleftarrow{\mathrm{R}} \mathcal{K}$, samples a random bit $b \in \{0,1\}$, initializes the set $Q := \emptyset$, and runs $\mathcal{A}(1^\lambda)$.
- **Pre-challenge queries:** For each query (ϕ, x), the challenger computes $y \leftarrow F_{\phi(k)}(x)$, sends y to $\mathcal{A}$, and proceeds to update $Q \leftarrow Q \cup \{(\phi, x)\}$.
- **Challenge query:** $\mathcal{A}$ sends challenge query $(\phi_0^*, \phi_1^*, x^*)$, *subject to the restriction that* $(\phi_c^*, x^*) \notin Q$, $\forall c \in \{0,1\}$. The challenger computes $y^* \leftarrow F_{\phi_b^*(k)}(x^*)$ and sends y^* to $\mathcal{A}$.
- **Post-challenge queries:** For each query (ϕ, x) *subject to the restriction that* $(\phi, x) \neq (\phi_c^*, x^*), \forall c \in \{0,1\}$, the challenger computes $y \leftarrow F_{\phi(k)}(x)$, and sends y to $\mathcal{A}$.
- **Guess:** $\mathcal{A}$ outputs its guess b'.

$\mathcal{A}$ wins if $b' = b$ and its advantage is defined as $|\Pr[\mathcal{A}$ wins$] - \frac{1}{2}|$, where the probability is over the internal coins of $\mathcal{A}$ and choice of k.

The lemma follows immediately from a standard hybrid argument. By RKA-security of the PRF F we have that $F_{\phi_0(k)}(x) \approx_c R(x) \approx_c F_{\phi_1(k)}(x)$, where R is a random function. Therefore, a distinguisher would directly contradict the security of the RKA-PRF. ■

5.2 DDH-Based Construction

In this section, we describe the DDH-based RKA-secure PRF construction of Bellare and Cash [7] (later extended by Abdalla et al. [1]) and describe how it fits into Construction 2 to realize a DDH-based CPRF for inner-product predicates.

RKA-Secure PRF from DDH. The multiplicative variant [1,7] of the Naor-Reingold PRF [53] is parameterized by an integer $n \geq 1$ and a multiplicative group $\mathbb{G}$ of prime order p with generator g. The PRF key $\mathbf{k} = (a_1, \ldots, a_n) \in \mathbb{Z}_p^n$ consists of n random elements in $\mathbb{Z}_p^n$ and the input $x \in \{0,1\}^n \setminus \{0^n\}$ is chosen from the set of all non-zero n-bit strings. The PRF NR^* is then defined as:

$$\mathsf{NR}^*((a_1, \ldots, a_n), x) = g^{\prod_{i=1}^n a_i^{x_i}}. \tag{1}$$

The RKA-secure version of the multiplicative Naor-Reingold PRF is parameterized by a collision-resistant hash function $h \colon \{0,1\}^n \times \mathbb{G}^n \to \{0,1\}^{n-2}$ and is defined as:[4]

$$\mathsf{NR}^*((a_1, \ldots, a_n), 11 \| h(x, g^{a_1}, \ldots, g^{a_n})). \tag{2}$$

[4] Note that the prefix "11" ensures that the input is never 0^n, and therefore always in the domain of NR^* [1,7].

Abdalla et al. [1, Section 4] show that Eq. (2) is an RKA-secure PRF for Φ_{aff}-restricted adversaries. We provide an informal merger of the main theorems from Abdalla et al. [1] pertaining to this construction here, for completeness.

Proposition 1 (Merge of [1, Theorems 4.5, 5.1, & A1]). *Let $\mathbb{G}$ be a multiplicative group of prime order p and let NR^* be defined as in Eq. (1). Let $h\colon \{0,1\}^n \times \mathbb{G}^n \to \{0,1\}^{n-2}$ be a collision-resistant hash function. Define the PRF family $\mathcal{F} = \{F_{\mathbf{k}}\colon \{0,1\}^n \to \mathbb{G}\}_{\mathbf{k}\in\mathbb{Z}_p^n}$ to be as in Eq. (2). Then, if the DDH assumption holds in $\mathbb{G}$, $\mathcal{F}$ is RKA-secure against all efficient, Φ_{aff}-restricted adversaries $\mathcal{A}$.*

Remark 5 (RKA security under DDH). Abdalla et al. [1] prove the RKA security of their construction for Φ_{aff}-restricted adversaries under the 1-DDHI assumption (which is known to be equivalent to the Square DDH assumption [10]). However, they explicitly note that, by combining Theorems 4.5, 5.1, & A1 (found in the full version of their paper), they obtain the same result under the DDH assumption. This same result was also used by Attrapadung et al. [3].

Remark 6 (Supporting vector inputs). NR^* takes as input a *binary* string $x \in \{0,1\}^n$ as opposed to a vector $\mathbf{x} \in \mathbb{F}^\ell$ as is assumed by our framework. However, we can easily map any $\mathbf{x} \in \mathbb{F}^\ell$ to a binary string of required length via any collision-resistant hash function (CRHF), which are known from the discrete logarithm assumption [35] (implied by DDH), making vector inputs $\mathbf{x} \in \mathbb{F}^\ell$ syntactically cleaner and without any loss of generality. In particular, for a CRHF h, the binary string input x can be computed as $h(\mathbf{x})$. Moreover, since the RKA-secure variant of NR^* already requires hashing the input using a CRHF, this does not introduce additional computational complexity.

Construction from DDH-Based RKA-Secure PRF. With the RKA-secure PRF construction of Proposition 1, we can instantiate Construction 2. To satisfy the key space and related-key derivation requirements, we must instantiate our extended framework with the following parameters. Let p be the order of the DDH-hard group $\mathbb{G}$. We set $\mathbb{F}$ to be a field extension of $\mathbb{F}_p$, and let $n = n(\lambda) \in \mathsf{poly}(\lambda)$, following Eq. (2). Applying Theorem 2 in conjunction with Proposition 1 yields:

Theorem 3. *Assume that the DDH assumption holds in a cyclic group $\mathbb{G}$ of order p. Then, there exists a (1-key, selectively-secure, constraint-hiding) CPRF for inner-product constraint predicates with vectors in $\mathbb{F}_p^\ell$, for any $\ell \geq 1$.*

Remark 7 (Complexity of the DDH-based construction). The Naor-Reingold PRF from Eq. (1) can be evaluated in NC^1. Interestingly, the same is true of the RKA-secure variant of Eq. (2), provided that the collision resistant hash function can be evaluated in NC^1 (which is the case of the discrete log based construction [35]).

5.3 VDLPN-Based Construction

In this section, we show that we can instantiate Construction 2 from any RKA-secure PRF supporting only additive key derivation functions $\varPhi_+ \subset \varPhi_{\mathsf{aff}}$ over the field $\mathbb{F}_2$. In particular, this allows us to instantiate our framework using the weak PRF candidate of Boyle et al. [23] based on the Variable-Density Learning Parity with Noise (VDLPN) assumption. This yields the first construction of a (weak) CPRF for inner-product predicates under a code-based assumption.

RKA-Secure weak PRF Candidate from VDLPN. For a security parameter λ, the VDLPN-based weak PRF candidate of Boyle et al. [23] is parameterized by integers $D = D(\lambda)$, $w = w(\lambda)$, input space $\{0,1\}^n$ and key space $\{0,1\}^n$, where $n := w \cdot D(D-1)/2$. The PRF F_K is defined as:

$$F_K(x) = \bigoplus_{i=1}^{D} \bigoplus_{j=1}^{w} \bigwedge_{k=1}^{i} (K_{i,j,k} \oplus x_{i,j,k}). \tag{3}$$

Theorem 4 (Informal; adapted from [23, Theorem 6.9]). *Let λ be a security parameter and suppose that the VDLPN assumption holds with parameters $w(\lambda)$ and $D(\lambda)$. Then, the PRF in Eq. (3) is an RKA-secure weak PRF with respect to additive key derivation functions $\varPhi_+$.*

We will use the following lemma which proves that for the case of $\mathbb{F}_2$, additive and affine RKA security are in fact equivalent in our context:

Lemma 2. *Let F be a PRF with key space $\mathbb{F}_2^n$ that is secure against $\varPhi_+$-restricted adversaries. Then, Construction 2 instantiated with F is a secure CPRF.*

Proof. Consider the proof of Theorem 2. We look at the queries issued by the CPRF challenger to the RKA oracle $\mathcal{O}_{\mathsf{rka}}$ in Hybrid $\mathcal{H}_2$ of the proof. For each query $\mathbf{x}$ issued by the adversary to the CPRF challenger, the induced affine function $\phi \in \varPhi_{\mathsf{aff}}$ is parameterized by vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_2^n$. Note that $\mathbf{a} = (a_1, \ldots, a_n)$, where $a_i \leftarrow \langle \mathbf{z}, \mathbf{x} \rangle$. Moreover, $a_i \neq 0$ for all queries that do not satisfy the constraint, which implies that $a_i = 1 \in \mathbb{F}_2$. As such, each (constrained) query issued to the RKA oracle $\mathcal{O}_{\mathsf{rka}}$ by the challenger is an affine function $\phi \in \varPhi_{\mathsf{aff}}$ parameterized by $(\mathbf{1}, \mathbf{b})$ and the oracle $\mathcal{O}_{\mathsf{rka}}$ responds with the PRF evaluated using key $\mathbf{k} := \mathbf{1} \circ \varDelta + \mathbf{b}$. This is equivalent to an *additive* function $\phi' \in \varPhi_+$ simply parameterized by $\mathbf{b}$. The reduction in Theorem 2 therefore goes through as before, concluding the lemma. ■

Construction from VDLPN-Based RKA-Secure weak PRF. With the RKA-secure weak PRF construction of Eq. (3), we can instantiate Construction 2. To satisfy the key space and related-key derivation requirements, we must instantiate our extended framework with the following parameters. We set $\mathbb{F}$ to be a field extension of $\mathbb{F}_{2^n}$, $n = n(\lambda) \in \mathsf{poly}(\lambda)$, map maps from $\mathbb{F}$ to $\mathbb{F}_2^n$, and $\ell \geq n$ (inputs of length ℓ can be truncated to n before being fed into the PRF, without loss of generality). Applying Theorem 2 in conjunction with Theorem 4 and Lemma 2 yields:

Theorem 5. *Assume that the VDLPN assumption holds. Then, there exists a (1-key, selectively-secure, constraint-hiding) weak CPRF for inner-product constraint predicates computed over vectors in $\mathbb{F}_2^\ell$, where $\ell \geq n$.*

6 CPRFs for Inner-Product Predicates from OWFs

In this section, we instantiate our extended framework from Sect. 5.1 under the minimal assumption that one-way functions exist. Unlike our constructions in Sect. 5.1, here we will require that the set of possible related keys computed for evaluation queries is bounded by a fixed polynomial $t = t(\lambda)$, which forces us to restrict the input domain of the CPRF. Specifically, we show that we can satisfy this requirement *without* placing any restrictions on the CPRF adversary if the CPRF inputs are vectors in $[0, B)^\ell$ with $B \in O(1)$ and $\ell = \ell(\lambda) \in O(\log \lambda)$. These restrictions limit the L_∞-norm of each input vector and make the input domain of the CPRF polynomial in the security parameter. We note that this is the same class of inner-product constraints considered by Davidson et al. [36] (inner products over $\mathbb{Z}$) from the LWE assumption, albeit here we only obtain a polynomially-sized input domain.

Our construction builds off of a result by Applebaum and Widder [2], which constructs a restricted class of RKA-secure PRFs from any PRF and a m-wise independent hash function. Their construction is secure against *additive* relations over a group, provided that the RKA adversary uses at most $t = t(\lambda)$ different related-key derivation functions $\phi_1 \ldots, \phi_t \in \Phi_+$, where $t \ll m$. (We stress, however, that the adversary can query the PRF on an unbounded number of inputs using each of the t different RKA functions.) Because m-wise independent hash functions can be constructed unconditionally [63], the resulting RKA-secure PRF can be realized from any PRF, thus relying only on the assumption that one-way functions exist [2,42]. More formally, they prove:

Theorem 6 (Adapted from [2]). *Let $\mathcal{K} = \{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ be a sequence of efficiently computable additive groups, and $t = t(\lambda)$ be an arbitrary fixed polynomial. Then, assuming the existence of a PRF $\mathcal{F} = \{F_k : \mathcal{X}_\lambda \rightarrow \mathcal{Y}\}_{k \in \mathbb{G}_\lambda}$, there exists an RKA-secure PRF with respect to addition over $\mathcal{K}$ provided that the total number of unique related-key derivation functions queried by the adversary is bounded by t. (The adversary is allowed to query each function on any number of inputs.)*

Unfortunately, we require the PRF to be RKA-secure with respect to *affine* relations Φ_{aff} and therefore cannot apply Theorem 6 directly. More concretely, the issue with affine (as opposed to additive) relations is that they are not "claw-free," meaning that there exist pairs of different functions $\phi_1, \phi_2 \in \Phi_{\mathsf{aff}}$ such that for a key $k \in \mathcal{K}$, $\phi_1(k) = \phi_2(k)$. The lack of claw-freeness poses problems in security proofs because, if an adversary is able to find two different $\phi_1, \phi_2 \in \Phi_{\mathsf{aff}}$ such that $\phi_1(k) = \phi_2(k)$, the adversary learns information about k and can then break the RKA-security of the PRF [1]. To address this, we strengthen Theorem 6 for the case of Φ_{aff}-restricted adversaries by showing that the number of collisions is bounded by a negligible factor in the security parameter, proving a stronger theorem via their approach. We describe this next.

6.1 Affine RKA-Secure PRFs from OWFs

In this section, we show how to construct RKA-secure PRFs for affine related-key derivation functions from one-way functions. The framework and proof closely follows that of Applebaum and Widder [2] for constructing RKA-secure PRFs from m-wise independent hash functions.

Immunizing PRFs Against RKA. The idea of Applebaum and Widder [2] is to immunize any regular PRF family $\mathcal{F}$ with key space $\mathcal{K} = \mathcal{K}_\lambda$ against a *bounded* related-key attack, where the adversary makes at most t related key queries (but can make an unbounded number of PRF queries under each related key) for some apriori fixed $t = t(\lambda) \in \mathsf{poly}(\lambda)$. The high level idea is to use a long key s from a large key space $\mathcal{S}$ (larger than $\mathcal{K}^t$) and use a public hash function h to derive shorter key $h(s) \in \mathcal{K}$ for $\mathcal{F}$. Here, we generalize their approach to the case of affine functions.

Definition 7 (t-good hash function). *Let λ be a security parameter, $\mathbb{F}$ be finite field of order at least 2^λ, and $\mathcal{K} \subseteq \{0,1\}^\lambda$ be a set of strings. A hash function $h \colon \mathbb{F} \to \mathcal{K}$ is said to be t-good if for any t-tuple of distinct affine function $(\phi_1, \ldots, \phi_t) \in \Phi^t_{\mathsf{aff}}$, the joint distribution of $(h(s), h(\phi_1(s)), \ldots, h(\phi_t(s)))$ induced by a random choice of $s \xleftarrow{R} \mathbb{F}$, is ε-close in statistical distance to the uniform distribution over $\mathcal{K}^{t+1}$, for some negligible $\varepsilon = \varepsilon(\lambda)$.*

Definition 8 (t-good hash family). *Let λ be a security parameter, $\mathbb{F}$ be a finite field of order at least 2^λ, and $\mathcal{Z}, \mathcal{K} \subseteq \{0,1\}^\lambda$. A family of hash functions $H = \{h_z \colon \mathbb{F} \to \mathcal{K}\}_{z \in \mathcal{Z}}$ is said to be t-good if with all-but-negligible probability, for a randomly selected $z \xleftarrow{R} \mathcal{Z}$, the hash function h_z is t-good.*

Remark 8 (Relation to correlation-robustness). We note that a t-good hash function can instantiated via a suitable correlation-robust hash function (and, in particular, a random oracle), which provides an alternative strategy to constructing CPRFs (in the selective security regime) from a random oracle.

We now prove that if we have a t-good hash family, we can "immunize" any PRF against affine related key attacks. Later, in Lemma 3, we show how to construct a t-good hash family from m-wise independent hash functions.

Theorem 7 (Extended from [2, Lemma 7.1]). *Let λ be a security parameter, $t = t(\lambda) \in \mathsf{poly}(\lambda)$, $\mathbb{F}$ be a finite field of order at least 2^λ, and $\mathcal{Z}, \mathcal{K} \subseteq \{0,1\}^\lambda$. Let $\mathcal{F} = \{F_k \colon \mathcal{X} \to \mathcal{Y}\}_{k \in \mathcal{K}}$ be a PRF family and $H = \{h_z \colon \mathbb{F} \to \mathcal{K}\}_{z \in \mathcal{Z}}$ be a t-good hash family. The PRF family $\mathcal{G} = \{G_{s,z} \colon \mathcal{X} \to \mathcal{Y}\}_{s \in \mathbb{F}, z \in \mathcal{Z}}$, parameterized by a secret $s \xleftarrow{R} \mathbb{F}$ and public $z \xleftarrow{R} \mathcal{Z}$, and defined by the mapping $G_{s,z}(x) \mapsto F_k(x)$, where $k \leftarrow h_z(s)$, is an RKA-secure PRF family against t-bounded Φ_{aff}-restricted adversaries.*

Proof. Suppose, towards contradiction, there exists an efficient Φ_{aff}-restricted $\mathcal{A}$ that has non-negligible advantage in the RKA-security game for G. Then, there exists a non-negligible function ν such that,

$$\left| \Pr_{s \overset{\mathrm{R}}{\leftarrow} \mathbb{F}, z \overset{\mathrm{R}}{\leftarrow} \mathcal{Z}} [\mathcal{A}^{G_{s,z}}(1^\lambda, z)] - \Pr_{z \overset{\mathrm{R}}{\leftarrow} \mathcal{Z}} [\mathcal{A}^{R}(1^\lambda, z)] \right| \geq \nu(\lambda),$$

where R is a truly random function.

Then, consider a vector of $t+1$ keys $\mathbf{k} := (k_0, k_1, \ldots, k_t) \in \mathcal{K}^{t+1}$, and define a stateful oracle $\mathcal{O}_{\mathbf{k}}$ as follows.

Oracle $\mathcal{O}_{\mathbf{k}}$

Initialize. Set $Q_\phi := \{\}$, define a dictionary $T := [\,]$, and counter $j \leftarrow 1$.

Evaluation.

- For each non-RKA query x, output $F_{k_0}(x)$.
- For each RKA query (ϕ, x):
  - If $\phi \in Q_\phi$, retrieve $k_i \leftarrow T[\phi]$ and output $F_{k_i}(x)$.
  - If $\phi \notin Q_\phi$, set $T[\phi] \leftarrow k_j$, set $j \leftarrow j+1$, and output $F_{k_j}(x)$.

In words, $\mathcal{O}_{\mathbf{k}}$ outputs $F_{k_i}(x)$, and stores the association between ϕ and k_i to answer all future queries involving ϕ using PRF key k_i.

Now, because h_z is t-good, for a random vector $\mathbf{k}$ of $t+1$ keys, we have that

$$\left| \Pr_{\mathbf{k} \overset{\mathrm{R}}{\leftarrow} \mathcal{K}^{t+1}, z \overset{\mathrm{R}}{\leftarrow} \mathcal{Z}} [\mathcal{A}^{\mathcal{O}_{\mathbf{k}}}(1^\lambda, z)] - \Pr_{z \overset{\mathrm{R}}{\leftarrow} \mathcal{Z}} [\mathcal{A}^{G_{s,z}}(1^\lambda, z)] \right| \geq \nu(\lambda) - \mathsf{negl}(\lambda).$$

By a straightforward hybrid argument, it follows that $\mathcal{A}$ has non-negligible advantage in winning the (standard) PRF game by distinguishing between $\mathcal{O}_{\mathbf{k}}$ and the truly random function R, contradicting that $\mathcal{F}$ is a PRF. This proves security against Φ_{aff}-restricted adversaries. ■

The following lemma shows that any $\Omega(\lambda \cdot t^2)$-wise independent hash function with a sufficiently large domain is t-good in the sense of Definition 7. Moreover, an m-wise independent hash function can be constructed unconditionally for any m (e.g., using a universal hash based on random polynomials [63]).

Lemma 3. *Let λ be a security parameter, $t = t(\lambda) \in \mathsf{poly}(\lambda)$, and H be a family of m-wise independent hash function with domain $S = \{S_\lambda\}$ and range $K = \{K_\lambda\}$ where $m \geq \lambda(3t+5)(t+1)$, $|K_\lambda| = 2^\lambda$, and $|S_\lambda| = 2^{\lambda(2t+6)}$. Then, H is a t-good family of hash function. In particular, for all but a $2^{-\lambda}$ fraction of the functions in H, the distribution of $h_z \overset{R}{\leftarrow} H$ is $2^{-0.99\lambda}$-close to uniform.*

Proof. The proof is deferred to the full version of the paper [60], as it closely follows the proof strategy of Applebaum and Widder [2, Proof of Lemma 7.2] for the case of additive key derivation functions. ■

6.2 CPRF Construction from OWFs

Using the RKA-secure PRF construction from Theorem 7, we can instantiate Construction 2 with $\mathbb{F} = \mathbb{F}_p$, for sufficiently large $p \geq 2^{\lambda(2t+6)}$ as required by Lemma 3, and $n \geq 1$. However, we must set the input vector domain to $[0, B)^\ell \subset \mathbb{Z}^\ell$ with the vector length ℓ such that $B^\ell \leq t$. Specifically, this ensures that the total number of unique inputs to the t-good hash when deriving affine keys is bounded by $t = t(\lambda) \in \mathsf{poly}(\lambda)$. To see this, note that there are B^ℓ possible values for the inner product $\langle \mathbf{z_0}, \mathbf{x}\rangle + \Delta \langle \mathbf{z}, \mathbf{x}\rangle$ given that $\mathbf{z}$ and $\mathbf{z_0}$ are fixed while $\mathbf{x} \in [0, B)^\ell$ is chosen by the adversary. Hence, we can simply let map be defined by applying n different t-good hash functions component-wise to derive the PRF key in K^n. Then, applying Theorem 2 in conjunction with Theorem 7 yields:

Theorem 8. *Let λ be a security parameter and fix a polynomial $t = t(\lambda) \in \mathsf{poly}(\lambda)$. Assume that one-way functions exist. Then, there exists a (1-key, selectively-secure, constraint-hiding) CPRF for inner-product constraint predicates with $\ell = \ell(\lambda) \in O(\log \lambda)$ and input vectors in the range $[0, B)$ for any constant B such that $B^\ell \leq t$.*

Proof. We recall the proof of Theorem 2, and in particular Hybrid $\mathcal{H}_2$. In the game defined by $\mathcal{H}_2$, for each query $\mathbf{x}$ issued by the CPRF adversary, the challenger derives the affine function ϕ parameterized by vectors $\mathbf{a}, \mathbf{b} \in \mathbb{F}_p^n$ where:

- $\mathbf{a} := (a_1, \ldots, a_n)$ with $a_i = \langle \mathbf{z}, \mathbf{x}\rangle$ for all $i \in [n]$.
- $\mathbf{b} := (b_1, \ldots, b_n)$ with $b_i = \langle \mathbf{z_0}_i, \mathbf{x}\rangle$ for all $i \in [n]$.

Note that $\mathbf{z}$ and $\mathbf{z_0}_i$, for all $i \in [n]$ are fixed at the start of the CPRF game. Therefore, $\mathbf{a}, \mathbf{b}$ are both entirely determined by the query vector $\mathbf{x}$. The RKA oracle $\mathcal{O}_{\mathsf{rka}}$ in $\mathcal{H}_2$ (when instantiated with the immunized RKA-PRF construction of Theorem 7) computes the RKA key as $h_i(a_i \Delta_i + b_i)$ for all $i \in [n]$, where h_i is an independent t-good hash function and Δ_i is an independent PRF key. We must show that, for all possible sets of queries $Q := \{\mathbf{x}_j \mid 1 \leq j \leq q_E\}$ issued by $\mathcal{A}$ (here q_E is an arbitrary upper bound on the total number of evaluation queries), the number of unique inputs to h_i never exceeds t. This follows from the fact that the number of possible values that $k_i := a_i \Delta_i + b_i$ can take on is bounded by the number of unique values of $\mathbf{x}$, which in turn is bounded by $B^\ell \leq t$, by construction. We stress that there are no restrictions placed on the adversary's queries—the adversary can adaptively query the CPRF challenger and issue any polynomial number of evaluation queries (independently of t). ■

As a corollary, we obtain an analogous result to Theorem 8 but with an exponential input domain provided that the CPRF adversary makes at most t unique evaluation queries on constrained inputs.

Corollary 1. *Let λ be a security parameter and fix a polynomial $t = t(\lambda) \in \mathsf{poly}(\lambda)$. Assume that one-way functions exist. Then, there exists a (1-key, selectively-secure, constraint-hiding) CPRF for inner-product constraint predicates for any $\ell \geq 1$ provided that the adversary makes at most t constrained evaluation queries.*

7 Evaluation

In this section, we implement[5] and benchmark our CPRF constructions. For each construction, we first analyze the complexity (in terms of multiplication, additions, and invocations of other cryptographic primitives) and then report the concrete performance of our Go (v1.20) implementation benchmarked on an Apple M1 CPU. All benchmarks are performed on a single core.

7.1 Complexity and Benchmarks

Random Oracle Construction. The random oracle construction requires computing the inner product in $\mathbb{F}$ followed by a call to a random oracle. We heuristically instantiate the random oracle using the SHA256 hash function. We let the $\mathbb{F} = \mathbb{F}_p$ be a finite field where p is a 128-bit prime. The bottleneck of the construction is computing the inner product (modulo p), which requires a total of ℓ modular multiplications and additions. We report the concrete performance in Table 2. Overall, evaluation requires a few microseconds of computation time, ranging from 2μs for small vectors ($\ell = 10$) and 200μs for large vectors ($\ell = 1000$).

DDH-Based Construction. In the DDH-based construction, the bulk of the required operations are performed modulo p, where p is the order of the DDH-hard group. For a security parameter λ and $n = n(\lambda)$, the CPRF construction requires computing (1) $n\ell$ multiplications and $n\ell$ additions (mod p) to compute the inner products between length-ℓ vectors, (2) one invocation of a collision-resistant hash function, and (3) n multiplications (mod p) and $n + 1$ group operations in $\mathbb{G}$ to compute the PRF evaluation. This results in a total complexity of $n(\ell+1)$ multiplications (mod p), $n\ell$ additions, $n+1$ group operations, and one invocation of a CRHF. Using the P256 elliptic curve, letting $n = 128$, and using the discrete logarithm based CRHF construction described in the full version [60], each CPRF evaluation requires a few milliseconds to compute (note that in practice, the DL-based CRHF can be replaced with a fixed-key AES or SHA256 hash function for better performance). We report the concrete performance in Table 3. The concrete performance is worse for smaller vectors due to constant overheads of computing the CRHF and PRF relative to computing the inner product. For larger vectors, however, the inner product computation dominates the cost.

OWF-Based Construction. Our OWF-based construction requires computing the inner products over the integers, which requires ℓ multiplications and ℓ additions in $\mathbb{Z}$ to compute inner products. Then, we need to evaluate an m-wise independent hash function. This requires evaluating a random polynomial of degree $m = O(\lambda \cdot t^2)$ with $O(\lambda \cdot t)$-bit coefficients (recall Lemma 3). Here, we let $\lambda = 40$ as it is a statistical security parameter of the t-good hash function. For very small values of B and ℓ, we obtain reasonable concrete efficiency when

[5] The implementation is open source: https://github.com/sachaservan/cprf.

Table 2. Concrete evaluation time for our RO-based CPRF construction for vectors of length ℓ.

(ℓ)	10	50	100	500	1000
	2 µs	10 µs	19 µs	98 µs	200 µs

Table 3. Concrete evaluation time for our DDH-based CPRF construction for vectors of length ℓ.

(ℓ)	10	50	100	500	1000
	8 ms	11 ms	16 ms	46 ms	85 ms

evaluating the m-wise independent hash function (less than one second of computation for $B = 2$ and $\ell = 5$ and roughy 50MB public parameters). However, for larger parameters, the concrete efficiency quickly becomes impractical. This blowup is due to the quadratic overhead of Lemma 3. Additionally, the public parameters quickly become impractically large (in the *petabytes*) as ℓ increases, due to the cubic factor in t description of the random polynomial. Indeed, this description already reaches terabytes in size with $B = 2$ and $\ell = 10$, barring any concretely practical instantiation.

7.2 Comparison to Other CPRF Constructions

Prior CPRF constructions for inner product (and NC^1) predicates [4,32,36] do not have implementations, and due to large parameters or heavy building blocks, are far too inefficient to be implemented. We briefly discuss the concrete efficiency roadblocks associated with these constructions.

- The LWE-based CPRF construction of Davidson et al. [36] is implementable but very inefficient due to the large parameters required for security and computationally expensive building blocks. Specifically, their construction requires computing a linear (in the input size) number of matrix-matrix products, which poses an efficiency roadblock. Similar roadblocks are faced with other LWE-based constructions, even if adapted to the simpler case of inner-product constraints. While concrete efficiency can be improved by assuming ring-LWE, the concrete costs remain high.
- The constructions of Attrapadung et al. [3] is tailored to evaluating NC^1 Boolean circuits and requires computing a linear number of group exponentiations in the degree of the universal NC^1 circuit computing the constraint predicate. While their construction can be theoretically applied to computing inner-product predicates, it does yield a practical solution given the need for emulating field operations *inside* of the NC^1 universal circuit.
- The approach of Couteau et al. [32] based on DCR requires evaluating a PRF using HSS (where the PRF key is encoded as an HSS input share). This requires evaluating a linear (in the degree of the polynomial computing the PRF) number of HSS multiplications. Using a DCR-based variant of the Naor-Reingold PRF necessitates computing $g^{\prod_i^n a_i^{x_i}}$ in HSS, where the key $k = (a_1, \ldots, a_n)$ is the PRF key provided as input. The exceedingly high degree of this polynomial eliminates the possibility of a concretely practical instantiation, since even low-degree polynomials can already be concretely expensive to evaluate in HSS schemes [20].

8 Conclusion and Future Work

In conclusion, this paper contributes a simple framework for constructing constraint-hiding CPRFs with inner-product constraint predicates through subtractive secret sharing and related-key-attack-secure PRFs. Through our framework, we constructed the first (1-key, selectively-secure, constraint-hiding) CPRFs with inner-product constraint predicates from DDH and from one-way functions, and the first (1-key, adaptively-secure, constraint-hiding) CPRFs in the random oracle model.

Finally, we note that recent work building pseudorandom correlation functions (PCFs) [27,33] (which have many applications in efficient multi-party computation) makes extensive use of CPRFs for inner-product predicates (and extensions thereof), including some of the constructions presented in this work. This motivates the further study of efficient CPRF instantiations, even for relatively simple predicates, from a wider range of assumptions.

Future Work. We identify several interesting avenues for future work. The first open problem is constructing (constraint-hiding) CPRFs for more expressive constraints from new assumptions, especially for NC^1 and puncturing constraints. Given the tight connection between our framework and RKA-secure PRFs, an additional avenue of exploration is constructing suitable RKA-secure PRFs from new assumptions (which will immediately enable instantiating our framework under those assumptions as well). Second, there are currently few practical applications of CPRFs with inner-product predicates that we are aware of (with the exception of recent constructions of PCFs [27,33]), which we believe is due to the previous lack of concretely efficient constructions. Finding additional practical use cases for CPRFs with inner-product predicates (whether constraint-hiding or not), is an interesting question and worth exploring further in light of our efficient constructions.

Acknowledgements. I'd like to thank Geoffroy Couteau for insightful discussions, providing me with several pointers and references (especially when it came to pointing out the relevance of [24,31]), and many invaluable suggestions. I'd like to thank Michele Orrù for editorial advice and feedback. I'm grateful to Vinod Vaikuntanathan and Yael Kalai for helpful discussion on early ideas surrounding this work.

Finally, I'm very thankful to the anonymous reviewers for pointing out a bug in an earlier theorem and construction, and providing many helpful comments that improved the presentation and readability of the paper.

References

[1] Michel Abdalla, Fabrice Benhamouda, Alain Passelègue, and Kenneth G Paterson. Related-key security for pseudorandom functions beyond the linear barrier. In *Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I 34*, pages 77–94. Springer, 2014.

[2] Benny Applebaum and Eyal Widder. Related-key secure pseudorandom functions: The case of additive attacks. *Cryptology ePrint Archive*, 2014.

[3] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Constrained PRFs for in traditional groups. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II*, pages 543–574. Springer, 2018.

[4] Nuttapong Attrapadung, Takahiro Matsuda, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively single-key secure constrained PRFs for NC^1. In *IACR International Workshop on Public Key Cryptography*, pages 223–253. Springer, 2019.

[5] Carsten Baum, Lennart Braun, Alexander Munch-Hansen, and Peter Scholl. Moz$\mathbb{Z}_{2^k}$arella: efficient vector-OLE and zero-knowledge proofs over $\mathbb{Z}_{2^k}$. In *Annual International Cryptology Conference*, pages 329–358. Springer, 2022.

[6] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In *Annual International Cryptology Conference*, pages 581–615. Springer, 2023.

[7] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In *Annual Cryptology Conference*, pages 666–684. Springer, 2010.

[8] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: RKA-PRPs, RKA-PRFs, and applications. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 491–506. Springer, 2003.

[9] Eli Biham. New types of cryptanalytic attacks using related keys. In *Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, EUROCRYPT '93, page 398–409, Berlin, Heidelberg, 1994. Springer-Verlag. ISBN 3540576002.

[10] Olivier Blazy and David Pointcheval. Traceable signature with stepping capabilities. In *Cryptography and Security: From Theory to Applications: Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday*, pages 108–131. Springer, 2012.

[11] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology-ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II 19*, pages 280–300. Springer, 2013.

[12] Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. *Algorithmica*, 79:1233–1285, 2017.

[13] Dan Boneh, Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Key homomorphic PRFs and their applications. In *Annual Cryptology Conference*, pages 410–428. Springer, 2013.

[14] Dan Boneh, Sam Kim, and Hart Montgomery. Private puncturable PRFs from standard lattice assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 415–445. Springer, 2017.

[15] Dan Boneh, Kevin Lewi, and David J Wu. Constraining pseudorandom functions privately. In *IACR International Workshop on Public Key Cryptography*, pages 494–524. Springer, 2017.

[16] Raphaël Bost, Brice Minaud, and Olga Ohrimenko. Forward and backward private searchable encryption from constrained cryptographic primitives. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1465–1482, 2017.

[17] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *International workshop on public key cryptography*, pages 501–519. Springer, 2014.

[18] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 337–367. Springer, 2015.

[19] Elette Boyle, Niv Gilboa, and Yuval Ishai. Breaking the circuit size barrier for secure computation under DDH. In *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 509–539. Springer, 2016.

[20] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, and Michele Orrù. Homomorphic secret sharing: optimizations and applications. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2105–2122, 2017.

[21] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III 39*, pages 489–518. Springer, 2019.

[22] Elette Boyle, Lisa Kohl, and Peter Scholl. Homomorphic secret sharing from lattices without FHE. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 3–33. Springer, 2019.

[23] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Correlated pseudorandom functions from variable-density LPN. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1069–1080. IEEE, 2020.

[24] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators from ring-LPN. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part II 40*, pages 387–416. Springer, 2020.

[25] Zvika Brakerski and Vinod Vaikuntanathan. Constrained key-homomorphic PRFs from standard lattice assumptions: Or: How to secretly embed a circuit in your PRF. In *Theory of Cryptography: 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II 12*, pages 1–30. Springer, 2015.

[26] Zvika Brakerski, Rotem Tsabary, Vinod Vaikuntanathan, and Hoeteck Wee. Private constrained PRFs (and more) from LWE. In *Theory of Cryptography Conference*, pages 264–302. Springer, 2017.

[27] Dung Bui, Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. Fast public-key silent OT and more from constrained Naor-Reingold. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 88–118. Springer, 2024.

[28] Ran Canetti and Yilei Chen. Constraint-hiding constrained PRFs for NC from LWE. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 446–476. Springer, 2017.

[29] Nishanth Chandran, Srinivasan Raghuraman, and Dhinakaran Vinayagamurthy. Reducing depth in constrained PRFs: From bit-fixing to NC^1. In *Public-Key Cryptography–PKC 2016*, pages 359–385. Springer, 2016.

[30] Yilei Chen, Vinod Vaikuntanathan, and Hoeteck Wee. GGH15 beyond permutation branching programs: proofs, attacks, and candidates. In *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II 38*, pages 577–607. Springer, 2018.

[31] Aloni Cohen, Shafi Goldwasser, and Vinod Vaikuntanathan. Aggregate pseudorandom functions and connections to learning. In *Theory of Cryptography: 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part II 12*, pages 61–89. Springer, 2015.

[32] Geoffroy Couteau, Pierre Meyer, Alain Passelègue, and Mahshid Riahinia. Constrained pseudorandom functions from homomorphic secret sharing. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 194–224. Springer, 2023.

[33] Geoffroy Couteau, Lalita Devadas, Srinivas Devadas, Alexander Koch, and Sacha Servan-Schreiber. Quietot: Lightweight oblivious transfer with a public-key setup. *Cryptology ePrint Archive*, 2024.

[34] Nan Cui, Shengli Liu, Yunhua Wen, and Dawu Gu. Pseudorandom functions from LWE: RKA security and application. In *Australasian Conference on Information Security and Privacy*, pages 229–250. Springer, 2019.

[35] Ivan Bjerre Damgård. Collision free hash functions and public key signature schemes. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 203–216. Springer, 1987.

[36] Alex Davidson, Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure constrained pseudorandom functions in the standard model. In *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17–21, 2020, Proceedings, Part I*, pages 559–589. Springer, 2020.

[37] Nico Döttling, Sanjam Garg, Yuval Ishai, Giulio Malavolta, Tamer Mour, and Rafail Ostrovsky. Trapdoor hash functions and their applications. In *Annual International Cryptology Conference*, pages 3–32. Springer, 2019.

[38] Thibauld Feneuil. *Post-Quantum Signatures from Secure Multiparty Computation.* PhD thesis, Sorbonne Université, 2023.

[39] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure functional encryption without obfuscation. *IACR Cryptol. ePrint Arch.*, 2014:666, 2014.

[40] Niv Gilboa and Yuval Ishai. Distributed point functions and their applications. In *Advances in Cryptology–EUROCRYPT 2014: 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings 33*, pages 640–658. Springer, 2014.

[41] David Goldenberg and Moses Liskov. On related-secret pseudorandomness. In *Theory of Cryptography Conference*, pages 255–272. Springer, 2010.

[42] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[43] Vipul Goyal, Adam O'Neill, and Vanishree Rao. Correlated-input secure hash functions. In *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*, pages 182–200. Springer, 2011.

[44] David Heath and Vladimir Kolesnikov. One hot garbling. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 574–593, 2021.

[45] Dennis Hofheinz, Akshay Kamath, Venkata Koppula, and Brent Waters. Adaptively secure constrained pseudorandom functions. In *International Conference on Financial Cryptography and Data Security*, pages 357–376. Springer, 2019.

[46] Susan Hohenberger, Venkata Koppula, and Brent Waters. Adaptively secure puncturable pseudorandom functions in the standard model. In *International conference on the theory and application of cryptology and information security*, pages 79–102. Springer, 2015.

[47] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In *Annual International Cryptology Conference*, pages 145–161. Springer, 2003.

[48] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 669–684, 2013.

[49] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *Automata, Languages and Programming: 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II 35*, pages 486–498. Springer, 2008.

[50] Arthur Lazzaretti and Charalampos Papamanthou. TreePIR: Sublinear-time and polylog-bandwidth private information retrieval from DDH. *Cryptology ePrint Archive*, 2023.

[51] Kevin Lewi, Hart Montgomery, and Ananth Raghunathan. Improved constructions of PRFs secure against related-key attacks. In *Applied Cryptography and Network Security: 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings 12*, pages 44–61. Springer, 2014.

[52] Yiping Ma, Ke Zhong, Tal Rabin, and Sebastian Angel. Incremental offline/online PIR. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1741–1758, 2022.

[53] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *Journal of the ACM (JACM)*, 51(2):231–262, 2004.

[54] Claudio Orlandi, Peter Scholl, and Sophia Yakoubov. The rise of Paillier: homomorphic secret sharing and public-key silent OT. In *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I 40*, pages 678–708. Springer, 2021.

[55] Chris Peikert and Sina Shiehian. Privately constraining and programming PRFs, the LWE way. In *IACR International Workshop on Public Key Cryptography*, pages 675–701. Springer, 2018.

[56] Naty Peter, Rotem Tsabary, and Hoeteck Wee. One-one constrained pseudorandom functions. In *1st Conference on Information-Theoretic Cryptography (ITC 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

[57] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on OT extension. *ACM Transactions on Privacy and Security (TOPS)*, 21(2):1–35, 2018.

[58] Kim Ramchen and Brent Waters. Fully secure and fast signing from obfuscation. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 659–673, 2014.

[59] Phillipp Schoppmann, Adrià Gascón, Leonie Reichert, and Mariana Raykova. Distributed vector-OLE: Improved constructions and implementation. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 1055–1072, 2019.

[60] Sacha Servan-Schreiber. Constrained pseudorandom functions for inner-product predicates from weaker assumptions. Cryptology ePrint Archive, Paper 2024/058, 2024. URL https://eprint.iacr.org/2024/058.

[61] Shi-Feng Sun, Xingliang Yuan, Joseph K Liu, Ron Steinfeld, Amin Sakzad, Viet Vo, and Surya Nepal. Practical backward-secure searchable encryption from symmetric puncturable encryption. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 763–780, 2018.

[62] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[63] Mark N Wegman and J Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of computer and system sciences*, 22(3):265–279, 1981.

Mild Asymmetric Message Franking: Illegal-Messages-Only and Retrospective Content Moderation

Zhengan Huang[1], Junzuo Lai[2](✉), Gongxian Zeng[1](✉), and Jian Weng[2]

[1] Pengcheng Laboratory, Shenzhen, China
gxzeng@cs.hku.hk

[2] College of Information Science and Technology, Jinan University, Guangzhou, China
tlaijunzuo@jnu.edu.cn

Abstract. Many messaging platforms have integrated end-to-end (E2E) encryption into their services. This widespread adoption of E2E encryption has triggered a technical tension between user privacy and illegal content moderation. The existing solutions either support only unframeability or deniability, or they are prone to abuse (the moderator can perform content moderation for all messages, whether illegal or not), or they lack mechanisms for retrospective content moderation.

To address the above issues, we introduce a new primitive called *mild asymmetric message franking* (MAMF) to establish illegal-messages-only and retrospective content moderation for messaging systems, supporting unframeability and deniability simultaneously. We provide a framework to construct MAMF, leveraging two new building blocks, which might be of independent interest.

Keywords: Asymmetric message franking · Set pre-constrained encryption · Hash proof system · Sigma protocol · Deniability and unframeability

1 Introduction

In recent years, there has been a substantial surge in the adoption of messaging applications deploying end-to-end (E2E) encryption, e.g., Facebook Messenger, WhatsApp and Signal, ensuring that the transmitted raw information cannot be obtained by the platforms.

Despite the security advantages, the widespread adoption of E2E encryption has not been universally welcomed. These encryption services might be misused for disseminating harmful content such as harassment messages, phishing links, fake news, and other potentially illegal information. Moreover, these services conflict with content moderation directly. Law enforcement and national security communities contend that such encryption hampers their ability to investigate, prosecute criminals and ensure public safety. In fact, the conflict between privacy and content moderation has spurred legislative proposals and policy campaigns about discouraging the deployment of E2E encryption [7,17].

On the other hand, technical experts have voiced concerns that these proposals, if implemented, might compromise the security provided by encryption systems [1], either by requiring unsafe alterations or prohibiting the use of E2E encryption altogether.

In recent years, many works have focused on employing cryptographic techniques to strike a balance between ensuring user privacy and effectively moderating illegal content within messaging systems.

One approach to uphold content moderation is through the use of message franking (MF) [5,6,10,18], seamlessly integrating with end-to-end (E2E) encryption, as discussed in [18]. This method empowers the receiver of a message to *report* it to a moderator (also referred to as the judge). In the MF framework, a valid report by the receiver includes the sender's identity, along with the message and specially constructed strings (e.g., signatures or hash values). These elements allow the moderator to verify whether the reported message originated from the identified sender. In order to strengthen user privacy, asymmetric message franking (AMF) [18] captures deniability. Informally, this property allows the sender to technically deny sending the message after a compromise, aiming to avoid potential backlash or embarrassment. AMF considers different scenarios regarding whose secret key is compromised. One scenario is judge compromise deniability, where a forger with the moderator's secret key can produce a signature indistinguishable from a real one. Unfortunately, existing works [12,13,18] *do not support unframeability.* Specifically, it becomes challenging for the moderator to convince law enforcement of the identity of the content originator, because judge compromise deniability allows the moderator to forge a message in the name of the sender. More importantly, current MF solutions do not adequately address a significant concern: the moderator can identify the sender of *all* reported messages, *regardless of the message's intent.* This extensive power of the moderator has the potential for abuse within the existing system.

Recently, a different proposal has emerged, suggesting end-to-end secure messaging with content moderation *exclusively for some pre-defined illegal contents*, based on set pre-constrained (SPC) group signatures [2]. However, as demonstrated in [2], their techniques are "tailored to obtain the strongest notion of unframeability and *no deniability*". Moreover, it remains unclear how to execute content moderation for newly identified illegal content that hasn't been predefined. This feature is reminiscent of "retrospective" access to encrypted data as considered in [9], which is in a somewhat different context and relies on extractable witness encryption [8].

In this paper, we aim to establish *mild* content moderation for a messaging system.

Firstly, we restrict the moderation capabilities to *illegal messages only*, while concurrently providing the capacity for *retrospective* content moderation. Notably, the set of illegal messages may increase over time. To carry out content moderation for the newly added illegal messages, a straightforward approach might involve system re-initialization; however, such a method impedes the moderator's ability to retrospectively moderate content. One objective of this paper is to enable the moderator to retrospectively examine past reports and identify those that qualify as illegal when new illegal messages are augmented.

Secondly, our content moderation system achieves trade-offs between deniability and unframeability. In [2], Bartusek et al. discuss the technical tension between deniability and unframeability, and claim that, while deniability [18] is a desirable property, it conflicts with unframeability. Upon meticulous examination, we discern that the primary reason why [18] cannot achieve unframeability is due to the fact that the scheme [18] supports judge-receiver compromise deniability. To elaborate, when the receiver's and the moderator's keys are compromised, a party possessing these keys can forge a signature that can be successfully accepted by both the receiver and the moderator. Our approach to achieve deniability and unframeability simultaneously imposes an additional constraint on deniability. Roughly, given any forged signature (for deniability), the receiver and the moderator can not accept it simultaneously. It then carves out space for unframeability, indicating that when both the receiver and the moderator identify the sender concurrently, the sender cannot deny sending the message.

1.1 Main Contributions

Our main contributions can be summarized as follows:

1. We introduce a new primitive called *mild asymmetric message franking* (MAMF) to establish mild content moderation for a messaging system, and formalize its security notions.
2. To construct MAMF, we introduce two new building blocks, *universal set pre-constrained encryption* (USPCE) and *dual hash proof system-based key encapsulation mechanism supporting Sigma protocols* (dual HPS-KEM$^{\Sigma}$), and present their concrete constructions.
3. We offer a generic framework of constructing MAMF from USPCE and dual HPS-KEM$^{\Sigma}$, and demonstrate that it fulfills the required security properties. By integrating a concrete USPCE scheme and a dual HPS-KEM$^{\Sigma}$ into the generic framework, we can obtain a concrete MAMF. We also have some improvements to enhance the efficiency of the concrete MAMF. Due to the space limitations, the improvements can be found in the full version of this paper.

MAMF Primitive. In the context of MAMF, four types of participants are involved: the sender, the receiver, the legislative agency, and the moderator (also referred to as the judge). MAMF comprises eleven algorithms: a setup algorithm Setup for generating global public parameters, three algorithms ($\mathsf{KG}_{\mathsf{Ag}}$, KG_{J}, KG_{u}) for generating key pairs, three algorithms (Frank, Verify, Judge) for creating and verifying genuine signatures, a token generation algorithm TKGen for retrospective content moderation, and three forging algorithms (Forge, RForge, JForge) for deniability.

We offer further explanations here.

Upon receiving the public parameter generated by Setup, the legislative agency selects a set S (representing illegal messages) and uses $\mathsf{KG_{Ag}}$ to generate a key pair for itself and an auxiliary parameter for the moderator. The moderator, leveraging the auxiliary parameter, invokes $\mathsf{KG_J}$ to create a key pair. The sender and receiver both utilize $\mathsf{KG_u}$ to generate their private/public keys.

The sender employs the franking algorithm Frank to generate a designated-verifier signature for a message m. The receiver utilizes Verify (with its secret key as input) to validate the received signature. If the received message is deemed illegal, and the receiver reports it to the moderator, the moderator can confirm the report using algorithm Judge, determining that the sender indeed sent the message. When the legislative agency intends to augment the set S with an additional illegal message for retrospective content moderation, it invokes the TKGen algorithm to produce a token for the new illegal message. With the aid of this token, the moderator can retrospectively examine past reports (as well as new reports). It's important to note that algorithms Forge, RForge, and JForge are not intended for execution by legitimate users. Their presence ensures deniability under specific compromise scenarios.

We address six distinct security requirements for MAMF: unforgeability, accountability, unframeability, deniability, untraceability, and confidentiality of sets.

1. *Unforgeability.* As the fundamental security prerequisite for general signatures, unforgeability in MAMF ensures prevention of successful impersonation, i.e., the receiver cannot be deceived into accepting a message not genuinely sent by the sender.
2. *Accountability.* Accountability ensures that the functionality of reporting illegal messages. In line with the definition in [13,18], accountability is formalized with two special properties: sender binding and receiver binding. Sender binding ensures that the sender cannot trick the receiver into accepting unreportable messages, and receiver binding ensures that the receiver cannot deceive the judge to frame an innocent sender.
3. *Deniability.* Deniability is formalized with three special properties: universal deniability, receiver compromise deniability, and judge compromise deniability. Universal deniability guarantees deniability when neither the receiver's secret key nor the judge's secret key is compromised. Receiver compromise deniability guarantees deniability when the receiver's secret key is compromised. Judge compromise deniability is formalized to guarantee deniability when the judge's secret key is compromised.
4. *Unframeability.* Unframeability of MAMF requires that no party, even given a receiver's secret key and the judge's secret key, is able to produce a signature acceptable to both the receiver and the judge. This property implies that once both the receiver and the judge identify the originator of some illegal message, they can generate an evidence (e.g., a NIZK proof) to convince the other party of the originator of the message.

5. *Untraceability.* Ensuring untraceability restricts the capabilities of both the legislative agency and the judge, thereby enhancing sender privacy. This concept formalizes into two distinct notions: untraceability against legislative agency and untraceability against judge. Untraceability against legislative agency guarantees that the agency cannot determine if someone has actually sent a message, no matter whether it is in the set of illegal message or not. Untraceability against judge ensures that, without the assistance of the legislative agency, the moderator cannot ascertain the sender's identity when the message is not in the set of illegal messages.
6. *Confidentiality of Sets.* Confidentiality of sets requires that the legislative agency's public key and the judge's public key will not disclose any information about the set of illegal messages (which should not be disclosed to the public, e.g., child sexual abuse material).

Analogous to AMF [18], MAMF can be integrated with E2E encryption, which guarantees the confidentiality of messages. So we do not consider confidentiality of messages for MAMF. Furthermore, our MAMF could be extended to accommodate group communications like [13]. We leave it as a future work.

Technical Overview. For MAMF construction, we introduce two new primitives, USPCE and dual HPS-KEM$^{\Sigma}$, and utilize them to show a framework of constructing MAMF. We provide a technical overview here.

USPCE. In [2], Bartusek et al. formulate set pre-constrained encryption (SPCE). Generally, SPCE requires the generation of a public/secret key pair for a predefined (illegal message) set S. In SPCE, decryption of a ciphertext, produced by encrypting a message with the public key and an item x, is only possible when $x \in \mathsf{S}$. If $x \notin \mathsf{S}$, the secret key holder gains no information about the message.

Note that SPCE is insufficient for constructing MAMF, primarily due to its inability to handle ciphertexts produced by encrypting messages with respect to items where $x \notin \mathsf{S}$, while in the MAMF framework, in order to carry retrospective content moderation, the moderator should be able to handle the messages not in the set S, as long as the legislative agency has provided the corresponding tokens. Henceforth, we introduce a primitive, called universal set pre-constrained encryption (USPCE), to address these challenges.

A USPCE comprises five key algorithms: $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{TKGen}, \mathsf{Dec})$, where two kinds of entities, the authority and users, are involved. The setup algorithm Setup, executed by the authority, takes the security parameter and a pre-defined set S as input, and outputs public parameters, an auxiliary parameter, and a master secret key. The users invoke KG with the public and auxiliary parameters to generate their key pairs. The encryption algorithm Enc takes a public key, an item x, and a message m as input, producing a ciphertext.

– If the item x belongs to the set S, the user can directly employ the decryption algorithm Dec (with their secret key as input) to output the message m.
– If $x \notin \mathsf{S}$, the authority can execute the token generation algorithm TKGen (with the master secret key as input) to create a token tk for the item x. Subsequently,

the user can utilize the decryption algorithm Dec, taking their secret key, the ciphertext and the token tk as input, to recover the message m.

It is required that there is a Sigma protocol to prove that the ciphertext is well-formed.

For USPCE, we require the following security properties.

– *Confidentiality against authority:* It is required that the authority cannot obtain meaningful information about the message from a ciphertext, no matter whether the item x belongs to the set S or not.
– *Confidentiality against users:* It is required that, without the token for an item $x \notin \mathsf{S}$ given by the authority, any user cannot obtain meaningful information about the message from a ciphertext associated with x.
– *Confidentiality of sets:* It is required that the public parameters and a user's public key will not disclose any information about the pre-defined set S.

A concrete construction of USPCE based on the DBDH assumption is provided in Sect. 4.

Dual HPS-KEM$^{\Sigma}$. We introduce another building block, called dual hash proof system-based key encapsulation mechanism supporting Sigma protocols (dual HPS-KEM$^{\Sigma}$), which roughly can be seen as a dual version of the HPS-KEM$^{\Sigma}$ proposed in [13].

In essence, in a dual HPS-KEM$^{\Sigma}$, ciphertexts are generated in accordance with the original HPS-KEM$^{\Sigma}$ approach, while encapsulated keys are created in two modes: one follows the original HPS-KEM$^{\Sigma}$ method, and the other adopts an extended version of HPS-KEM$^{\Sigma}$ where an additional tag t is included as input during the computation of the encapsulated key. Moreover, in dual HPS-KEM$^{\Sigma}$, two additional algorithms are required for the uniform sampling of encapsulated keys: one with a tag as input and the other without using a tag.

Expanding on this, a dual HPS-KEM$^{\Sigma}$ scheme consists of ten algorithms: Setup, KG, $\mathsf{Encap}_{\mathrm{c}}$, $\mathsf{Encap}_{\mathrm{k}}$, Decap, $\mathsf{Encap}^{*}_{\mathrm{c}}$, $\mathsf{dEncap}_{\mathrm{k}}$,$\mathsf{dDecap}$,$\mathsf{SamEncK}$ and $\mathsf{dSamEncK}$.

We start by concentrating on the first six algorithms, which comprise an ordinary HPS-KEM$^{\Sigma}$ scheme. Specifically, Setup generates the public parameter, and KG produces a pair of public/secret user keys. Given the public parameter, but without user's public key, $\mathsf{Encap}_{\mathrm{c}}$ outputs a well-formed ciphertext, and $\mathsf{Encap}^{*}_{\mathrm{c}}$ outputs a ciphertext that could be either well-formed or ill-formed. The algorithm $\mathsf{Encap}_{\mathrm{k}}$, sharing the same randomness space with $\mathsf{Encap}_{\mathrm{c}}$, takes the public parameter and a public key as input, and outputs an encapsulated key. Utilizing the secret key, the algorithm Decap decapsulates the ciphertexts to obtain the encapsulated keys. Correctness requires that given a ciphertext output by $\mathsf{Encap}_{\mathrm{c}}$ with randomness r, Decap will return an encapsulated key equal to that created by $\mathsf{Encap}_{\mathrm{k}}$ with the same randomness r.

The following properties inherited from HPS-KEM$^{\Sigma}$ are required:

1. *Universality:* Given a public key, it is difficult for any unbounded adversary without the corresponding secret key to generate an ill-formed ciphertext c,

an encapsulated key k, and randomness r_{c}^* (indicating that c is generated via $\mathsf{Encap}_{\mathsf{c}}^*$ with randomness r_{c}^*), such that with the ciphertext c as input, Decap outputs a key equal to k.
2. *Ciphertext unexplainability:* It is difficult to generate a ciphertext c and randomness r_{c}^* (indicating that c is generated via $\mathsf{Encap}_{\mathsf{c}}^*$ with randomness r_{c}^*), such that c is well-formed.
3. *Indistinguishability:* The ciphertext output by $\mathsf{Encap}_{\mathsf{c}}^*$ should be indistinguishable from the well-formed ciphertext output by $\mathsf{Encap}_{\mathrm{c}}$.
4. *SK-second-preimage resistance:* Given a pair of public/secret keys, it is difficult to generate another valid secret key for this public key.
5. *Smoothness:* For any fixed public key, the algorithm Decap, fed with a ciphertext generated via $\mathsf{Encap}_{\mathsf{c}}^*$ and a secret key randomly sampled from the set of secret keys corresponding to the public key, will output a key uniformly distributed over the encapsulated key space.

Now, let's shift our focus to the last four algorithms of dual HPS-KEM$^{\Sigma}$, i.e., $\mathsf{dEncap}_{\mathrm{k}}, \mathsf{dDecap}, \mathsf{SamEncK}$ and $\mathsf{dSamEncK}$.

The algorithm $\mathsf{dEncap}_{\mathrm{k}}$, sharing the same randomness space and the same encapsulated key space with $\mathsf{Encap}_{\mathrm{k}}$, takes the public parameter, a public key and a tag as input, and outputs an encapsulated key. Utilizing the secret key and the tag, the algorithm dDecap decapsulates the ciphertexts to obtain the encapsulated keys. Correctness requires that given a tag t and a ciphertext output by $\mathsf{Encap}_{\mathrm{c}}$ with randomness r, dDecap will return an encapsulated key equal to that generated by $\mathsf{dEncap}_{\mathrm{k}}$ using the same tag t and randomness r.

The algorithms $\mathsf{SamEncK}$ and $\mathsf{dSamEncK}$ are both used to uniformly sample encapsulated keys. In particular, $\mathsf{SamEncK}$ takes the public parameter as input, and outputs an encapsulated key, while $\mathsf{dSamEncK}$ takes both the public parameter and a tag as input, and outputs an encapsulated key.

The following properties are also required for dual HPS-KEM$^{\Sigma}$:

6. *Extended universality:* Given a public key, it is difficult for any unbounded adversary without the corresponding secret key to generate an ill-formed ciphertext c, an encapsulated key k, a tag t, and randomness r_{c}^* (indicating that c is generated via $\mathsf{Encap}_{\mathsf{c}}^*$ with randomness r_{c}^*), such that with the ciphertext c and the tag t as input, dDecap outputs a key equal to k.
7. *Key unexplainability:* Given a pair of public/secret keys, it is difficult to generate $(c, r_{\mathsf{c}}^*, k, r_{\mathsf{k}}^*)$ (where c is a ciphertext generated via $\mathsf{Encap}_{\mathsf{c}}^*$ using randomness r_{c}^*, and k is an encapsulated key generated via $\mathsf{SamEncK}$ using randomness r_{k}^*), such that k is the result of decapsulating c by Decap.
8. *Extended key unexplainability:* Given a pair of public/secret keys, it is difficult to generate $(c, r_{\mathsf{c}}^*, k, t, r_{\mathsf{k}}^*)$ (where c is a ciphertext generated via $\mathsf{Encap}_{\mathsf{c}}^*$ using randomness r_{c}^*, and k is an encapsulated key generated via $\mathsf{dSamEncK}$ using tag t and randomness r_{k}^*), such that k is the result of decapsulating c by dDecap using tag t.
9. *Extended smoothness:* For any fixed public key, the algorithm dDecap, fed with a ciphertext generated via $\mathsf{Encap}_{\mathsf{c}}^*$, a random tag, and a secret key randomly sampled from the set of secret keys corresponding to the public key, will output a key uniformly distributed over the encapsulated key space.

10. *Special extended smoothness:* For any fixed public/secret key pair, the algorithm dDecap, fed with a ciphertext generated via $\mathsf{Encap}^*_{\mathrm{c}}$, a secret random tag, and the fixed secret key, will output a key uniformly distributed over the encapsulated key space.

Consistent with [13], we require that there exist Sigma protocols to prove that some results are precisely output by KG, $\mathsf{Encap}_{\mathrm{c}}$, $\mathsf{Encap}_{\mathrm{k}}$, $\mathsf{Encap}^*_{\mathrm{c}}$, $\mathsf{dEncap}_{\mathrm{k}}$, $\mathsf{SamEncK}$ and $\mathsf{dSamEncK}$.

A concrete construction of dual HPS-KEM$^{\Sigma}$ based on the DDH assumption is provided in Sect. 5. Similar to [13], our dual HPS-KEM$^{\Sigma}$ construction can also be extended to be based on the k-linear assumption [11,16].

An MAMF Framework. Now, we briefly outline the generic construction of an MAMF from USPCE and dual HPS-KEM$^{\Sigma}$. The main idea is as follows.

Here, Setup algorithm directly invokes the setup algorithm of $\mathsf{dHPS\text{-}KEM}^{\Sigma}$, $\mathsf{KG}_{\mathsf{Ag}}$ calls the setup algorithm of USPCE, KG_{J} invokes the key generation algorithms of $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ and USPCE (e.g., $pk_{\mathsf{J}} = (pk'_{\mathsf{J}}, pk_{\mathsf{USPCE}})$ where pk'_{J} is output by the key generation algorithm of $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ and pk_{USPCE} is output by the key generation algorithm of USPCE), while KG_{u} solely calls the key generation algorithm of $\mathsf{dHPS\text{-}KEM}^{\Sigma}$.

The algorithm Frank, executed by the sender to generate an MAMF signature for a message m, proceeds as follows. It utilizes $\mathsf{Encap}_{\mathrm{c}}$ to generate a well-formed encapsulated ciphertext c, and then employs $\mathsf{Encap}_{\mathrm{k}}$ to generate an encapsulated key k_{r} for the receiver and $\mathsf{dEncap}_{\mathrm{k}}$ to generate k_{J} (associated with a randomly chosen tag t) for the judge, where $\mathsf{Encap}_{\mathrm{c}}$, $\mathsf{Encap}_{\mathrm{k}}$ and $\mathsf{dEncap}_{\mathrm{k}}$ use the same randomness r. Following this, it calls the encryption algorithm of USPCE to encrypt the tag t with randomness r_{USPCE}, using the message m as the item, to obtain a ciphertext c_{t}. Finally, it outputs a signature $\sigma = (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathrm{t}})$, where π is a NIZK proof (generated with witness $(sk_{\mathsf{s}}, t, r, \perp, \perp, r_{\mathsf{USPCE}})$) for the relation $\mathcal{R}$ in Fig. 1.

In the verification process (i.e., the algorithm Verify), the receiver confirms the signature's validity by checking (i) if the NIZK proof is valid, and (ii) if the decapsulated key, produced by decapsulating c via Decap, matches the key k_{r} provided in the signature.

In the moderation process (i.e., the algorithm Judge), if m is in the illegal message set, or the legislative agency have provided a token (by TKGen) for m to implement retrospective content moderation, the judge first decrypts c_{t} with the decryption algorithm of USPCE (with item m) to obtain a tag t. Then, he/she checks (i) if the NIZK proof is valid, and (ii) if the decapsulated key, produced by decapsulating c with tag t via dDecap, matches the key k_{J} provided in the signature.

In the token generation process (i.e., the algorithm TKGen), the legislative agency directly invokes the token generation algorithm of USPCE.

Now, let's shift our focus to the forging algorithms Forge, RForge and JForge. The relation $\mathcal{R}$ in Fig. 1 plays a pivotal role in the forging algorithms. Observe that the relation comprises three sub-relations connected by "OR" operations.

$$
\begin{array}{l}
\mathcal{R} = \{((\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathrm{t}}, m), (sk_{\mathsf{s}}, t, r, r^*_{\mathsf{c}}, r^*_{\mathsf{k}}, r_{\mathsf{USPCE}})) : \\
\quad (\ \ (pk_{\mathsf{s}}, sk_{\mathsf{s}}) \in \mathcal{R}_{\mathrm{s}} \\
\qquad \wedge (\ ((c, k_{\mathsf{J}}, pk'_{\mathsf{J}}), (t, r)) \in \mathcal{R}^{\mathsf{d}}_{\mathrm{c,k}} \ \wedge_{\mathrm{eq}} \ ((pk_{\mathsf{USPCE}}, m, c_{\mathrm{t}}), (t, r_{\mathsf{USPCE}})) \in \mathcal{R}_{\mathsf{ct}} \ ) \ \ ) \\
\quad \vee (\ \ (c, r^*_{\mathsf{c}}) \in \mathcal{R}^*_{\mathsf{c}} \ \wedge \ (k_{\mathsf{r}}, r^*_{\mathsf{k}}) \in \mathcal{R}^*_{\mathsf{k}} \ \wedge \ ((pk_{\mathsf{USPCE}}, m, c_{\mathrm{t}}), (t, r_{\mathsf{USPCE}})) \in \mathcal{R}_{\mathsf{ct}} \ \ ) \\
\quad \vee(\ (c, r^*_{\mathsf{c}}) \in \mathcal{R}^*_{\mathsf{c}} \wedge ((k_{\mathsf{J}}, (t, r^*_{\mathsf{k}})) \in \mathcal{R}^{\mathsf{d}*}_{\mathsf{k}} \wedge_{\mathrm{eq}} ((pk_{\mathsf{USPCE}}, m, c_{\mathrm{t}}), (t, r_{\mathsf{USPCE}})) \in \mathcal{R}_{\mathsf{ct}} \ ) \ )\}
\end{array}
$$

Fig. 1. Relation $\mathcal{R}$ for MAMF, where $\mathcal{R}_{\mathrm{s}}$ is a relation proving the validity of the sender's public/secret keys, $\mathcal{R}^{\mathsf{d}}_{\mathrm{c,k}}$ is a relation proving that (c, k_{J}) is generated via $\mathsf{Encap}_{\mathrm{c}}$ and $\mathsf{dEncap}_{\mathrm{k}}$ with the same randomness r, $\mathcal{R}^*_{\mathrm{c}}$ is a relation proving that c is a ciphertext output by $\mathsf{Encap}^*_{\mathrm{c}}$ with randomness r^*_{c}, $\mathcal{R}_{\mathsf{ct}}$ is a relation proving the USPCE ciphertext is well-formed, $\mathcal{R}^*_{\mathrm{k}}$ is to prove the encapsulated key of the receiver is generated via $\mathsf{SamEncK}$, and $\mathcal{R}^{\mathsf{d}*}_{\mathrm{k}}$ is to prove the encapsulated key of the judge is generated via $\mathsf{dSamEncK}$. Note that the symbol "$\wedge_{\mathrm{eq}}$" represents an "EQUAL-AND$_l$" operation between two relations, signifying that part (e.g., t) of the sub-witnesses in the relations are equal. The formal definition and further discussions are presented in the full version of this paper, due to the space limitations.

The first sub-relation is crafted for the sender, ensuring that the message is genuinely sent by the sender and convincing the receiver that the judge can successfully trace the originator once the message is reported. The second and the third sub-relations are devised for the forging algorithms to ensure deniability.

Specifically, the algorithm Forge first invokes $\mathsf{Encap}^*_{\mathsf{c}}$ with randomness r^*_{c} to generate an ill-formed encapsulated ciphertext c, and uniformly samples two encapsulated keys k_{r} and k_{J}, where k_{r} is sampled with $\mathsf{SamEncK}$ using randomness r^*_{k}. Following this, it uniformly chooses a tag t, and then calls the encryption algorithm of USPCE to encrypt t with randomness r_{USPCE}, using the message m as the item, to obtain a ciphertext c_{t}. Finally, it outputs a signature $\sigma = (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathrm{t}})$, where π is a NIZK proof (generated with witness $(\perp, t, \perp, r^*_{\mathsf{c}}, r^*_{\mathsf{k}}, r_{\mathsf{USPCE}})$) for the relation $\mathcal{R}$.

The algorithm RForge first invokes $\mathsf{Encap}^*_{\mathsf{c}}$ with randomness r^*_{c} to generate an ill-formed encapsulated ciphertext c, and computes an encapsulated key k_{r} by executing Decap to decapsulate c. Then, it chooses a random tag t, and samples k_{J} with $\mathsf{dSamEncK}$ using t and randomness r^*_{k}. Following this, it calls the encryption algorithm of USPCE to encrypt t with randomness r_{USPCE}, using the message m as the item, to obtain a ciphertext c_{t}. Finally, it outputs a signature $\sigma = (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathrm{t}})$, where π is a NIZK proof (generated with witness $(\perp, t, \perp, r^*_{\mathsf{c}}, r^*_{\mathsf{k}}, r_{\mathsf{USPCE}})$) for the relation $\mathcal{R}$.

The algorithm JForge first invokes $\mathsf{Encap}^*_{\mathsf{c}}$ with randomness r^*_{c} to generate an ill-formed encapsulated ciphertext c, and computes an encapsulated key k_{J} by executing dDecap (with a random tag t) to decapsulate c. Then, it samples k_{r} with $\mathsf{SamEncK}$ using randomness r^*_{k}. Following this, it calls the encryption algorithm of USPCE to encrypt t with randomness r_{USPCE}, using the message m as the item, to obtain a ciphertext c_{t}. Finally, it outputs a signature $\sigma = (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathrm{t}})$, where π is a NIZK proof (generated with witness $(\perp, t, \perp, r^*_{\mathsf{c}}, r^*_{\mathsf{k}}, r_{\mathsf{USPCE}})$) for the relation $\mathcal{R}$.

In summary, we have presented a generic construction of MAMF from USPCE and dual HPS-KEM$^{\Sigma}$. By incorporating a concrete USPCE and a concrete dual HPS-KEM$^{\Sigma}$, we can derive a specific instantiation of MAMF.

Security Analysis. We turn to show a high-level intuition that our MAMF framework achieves the required unforgeability, accountability, deniability, unframeability, untraceability, and confidentiality of sets.

Given the similarity in the security analysis of unforgeability and accountability, we will focus here on demonstrating how to achieve unforgeability.

Unforgeability requires that any adversary cannot generate a signature such that an honest receiver accepts it. Supposing that there is an adversary generating a signature $\sigma = (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}})$ such that an honest receiver accepts it, we have: (i) π is a valid proof for the relation $\mathcal{R}$, and (ii) $k_{\mathsf{r}} = \mathsf{Decap}(\mathsf{pp}, sk_{\mathsf{r}}, c)$. Observe that to generate the valid proof π for $\mathcal{R}$, the adversary needs to know witness $(sk_{\mathsf{s}}, t, r, \perp, \perp, r_{\mathsf{USPCE}})$ or $(\perp, t, \perp, r_{\mathsf{c}}^{*}, r_{\mathsf{k}}^{*}, r_{\mathsf{USPCE}})$.

- If the adversary knows $(sk_{\mathsf{s}}, t, r, \perp, \perp, r_{\mathsf{USPCE}})$, it implies that sk_{s} is a valid secret key of the sender. Since the adversary possesses no information about the sender's secret key beyond the knowledge of the sender's public key, it is contradictory to SK-second-preimage resistance of dual HPS-KEM$^{\Sigma}$.
- If the adversary knows $(\perp, t, \perp, r_{\mathsf{c}}^{*}, r_{\mathsf{k}}^{*}, r_{\mathsf{USPCE}})$, it implies that c is generated via $\mathsf{Encap}_{\mathsf{c}}^{*}$. The ciphertext unexplainability of dual HPS-KEM$^{\Sigma}$ guarantees that c is not well-formed with overwhelming probability. Thus, according to (ii), $(c, k_{\mathsf{r}}, r_{\mathsf{c}}^{*})$ leads to a successful attack on universality of dual HPS-KEM$^{\Sigma}$.

Now, we turn to analyze universal deniability, receiver compromise deniability, and judge compromise deniability within our MAMF framework. Given the similarity in the security analysis of these deniability aspects, we will focus solely on demonstrating how judge compromise deniability is achieved.

Judge compromise deniability requires that any adversary with the judge's secret key cannot distinguish between the outputs $\sigma = (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}})$ of Frank and JForge.

- Frank computes $(c \leftarrow \mathsf{Encap}_{\mathsf{c}}(\mathsf{pp}; r), k_{\mathsf{r}} \leftarrow \mathsf{Encap}_{\mathsf{k}}(\mathsf{pp}, pk_{\mathsf{r}}; r), k_{\mathsf{J}} \leftarrow \mathsf{dEncap}_{\mathsf{k}}(\mathsf{pp}, pk'_{\mathsf{J}}, t; r))$ with the same randomness r, where t is a random tag. On the other hand, JForge computes $c \leftarrow \mathsf{Encap}_{\mathsf{c}}^{*}(\mathsf{pp}; r_{\mathsf{c}}^{*})$ and $k_{\mathsf{r}} \leftarrow \mathsf{SamEncK}(\mathsf{pp}; r_{\mathsf{k}}^{*})$ with randomness r_{c}^{*} and r_{k}^{*}, respectively, and then decapsulates c by dDecap, using sk'_{J} and a random tag t, to obtain k_{J}.
  Note that for $c \leftarrow \mathsf{Encap}_{\mathsf{c}}(\mathsf{pp}; r)$, we obtain $k_{\mathsf{r}} = \mathsf{Encap}_{\mathsf{k}}(\mathsf{pp}, pk_{\mathsf{r}}; r) = \mathsf{Decap}(\mathsf{pp}, sk_{\mathsf{r}}, c)$ and $k_{\mathsf{J}} = \mathsf{dEncap}_{\mathsf{k}}(\mathsf{pp}, pk'_{\mathsf{J}}, t; r) = \mathsf{dDecap}(\mathsf{pp}, sk'_{\mathsf{J}}, t, c)$. The indistinguishability of dual HPS-KEM$^{\Sigma}$ guarantees that the tuple $(c, k_{\mathsf{r}}, k_{\mathsf{J}})$ output by Frank is indistinguishable from $(\widehat{c}, \widehat{k_{\mathsf{r}}}, \widehat{k_{\mathsf{J}}})$, where $\widehat{c} \leftarrow \mathsf{Encap}_{\mathsf{c}}^{*}(\mathsf{pp}; r_{\mathsf{c}}^{*})$, $\widehat{k_{\mathsf{r}}} = \mathsf{Decap}(\mathsf{pp}, sk_{\mathsf{r}}, \widehat{c})$ and $\widehat{k_{\mathsf{J}}} = \mathsf{dDecap}(\mathsf{pp}, sk'_{\mathsf{J}}, t, \widehat{c})$. Due to the smoothness of dual HPS-KEM$^{\Sigma}$, it guarantees that the tuple $(\widehat{c}, \widehat{k_{\mathsf{r}}}, \widehat{k_{\mathsf{J}}})$ is indistinguishable from $(\widehat{c}, \widetilde{k_{\mathsf{r}}}, \widehat{k_{\mathsf{J}}})$, where $\widetilde{k_{\mathsf{r}}}$ is uniformly distributed over the encapsulated key space. According to the uniformity of sampled key by algorithm $\mathsf{SamEncK}$ of dual HPS-KEM$^{\Sigma}$, $(\widehat{c}, \widetilde{k_{\mathsf{r}}}, \widehat{k_{\mathsf{J}}})$ is indistinguishable from that output by JForge.

Thus, the output tuple $(c, k_{\mathsf{r}}, k_{\mathsf{J}})$ from Frank and that from JForge are indistinguishable.

– The ciphertext c_{t} output by Frank and that output by JForge are distributed identically.
– Frank generates a NIZK proof π for relation $\mathcal{R}$ with witness $(sk_{\mathsf{s}}, t, r, \perp, \perp, r_{\mathsf{USPCE}})$, while JForge generates π for $\mathcal{R}$ with witness $(\perp, t, \perp, r_{\mathsf{c}}^*, r_{\mathsf{k}}^*, r_{\mathsf{USPCE}})$. The zero knowledge property of NIZK guarantees that anyone cannot distinguish the proof output by Frank from that output by JForge.

Unframeability requires that any adversary (possessing the secret keys of the receiver, the legislative agency and the judge, but without the sender's secret key) cannot generate a signature such that both the receiver and the judge accept it. Suppose that there is an adversary generating a signature $\sigma = (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathrm{t}})$ such that both the receiver and the judge accept it. The fact that the receiver accepts the signature implies: (i) π is a valid proof for the relation $\mathcal{R}$, and (ii) $k_{\mathsf{r}} = \mathsf{Decap}(\mathsf{pp}, sk_{\mathsf{r}}, c)$. Observe that to generate the valid proof π for $\mathcal{R}$, the adversary needs to know witness $(sk_{\mathsf{s}}, t, r, \perp, \perp, r_{\mathsf{USPCE}})$ or $(\perp, t, \perp, r_{\mathsf{c}}^*, r_{\mathsf{k}}^*, r_{\mathsf{USPCE}})$.

– If the adversary knows $(sk_{\mathsf{s}}, t, r, \perp, \perp, r_{\mathsf{USPCE}})$, it implies that sk_{s} is a valid secret key of the sender. Similar to the previous analysis of unforgeability, it is contradictory to SK-second-preimage resistance of dual HPS-KEM$^{\Sigma}$.
– If the adversary knows $(\perp, t, \perp, r_{\mathsf{c}}^*, r_{\mathsf{k}}^*, r_{\mathsf{USPCE}})$, we turn our focus on the last two sub-relations of relation $\mathcal{R}$.
  • If $(\perp, t, \perp, r_{\mathsf{c}}^*, r_{\mathsf{k}}^*, r_{\mathsf{USPCE}})$ satisfies

$$(c, r_{\mathsf{c}}^*) \in \mathcal{R}_{\mathsf{c}}^* \;\wedge\; (k_{\mathsf{r}}, r_{\mathsf{k}}^*) \in \mathcal{R}_{\mathsf{k}}^* \;\wedge\; ((pk_{\mathsf{USPCE}}, m, c_{\mathrm{t}}), (t, r_{\mathsf{USPCE}})) \in \mathcal{R}_{\mathsf{ct}},$$

  according to (ii), $(c, r_{\mathsf{c}}^*, k_{\mathsf{r}}, r_{\mathsf{k}}^*)$ leads to a successful attack on the key unexplainability of dual HPS-KEM$^{\Sigma}$.
  • If $(\perp, t, \perp, r_{\mathsf{c}}^*, r_{\mathsf{k}}^*, r_{\mathsf{USPCE}})$ satisfies

$$(c, r_{\mathsf{c}}^*) \in \mathcal{R}_{\mathsf{c}}^* \;\wedge\; (k_{\mathsf{J}}, (\boxed{t}, r_{\mathsf{k}}^*)) \in \mathcal{R}_{\mathsf{k}}^{\mathsf{d}*} \;\wedge_{\mathrm{eq}}\; ((pk_{\mathsf{USPCE}}, m, c_{\mathrm{t}}), (\boxed{t}, r_{\mathsf{USPCE}})) \in \mathcal{R}_{\mathsf{ct}},$$

  according to the fact that the judge accepts the signature (which further suggests $k_{\mathsf{J}} = \mathsf{dDecap}(\mathsf{pp}, sk'_{\mathsf{J}}, t, c)$), $(c, r_{\mathsf{c}}^*, k_{\mathsf{J}}, t, r_{\mathsf{k}}^*)$ leads to a successful attack on the extended key unexplainability of dual HPS-KEM$^{\Sigma}$.

Next, we turn to analyze untraceability against judge and untraceability against agency within our MAMF framework. Given the similarity in the security analysis of these untraceability aspects, we will focus solely on demonstrating how untraceability against judge is achieved.

Untraceability against judge requires the existence of a simulator SimFrank, such that any adversary with the judge's secret key cannot distinguish between the outputs $\sigma = (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathrm{t}})$ of Frank and SimFrank, given that the message is not in the set of illegal messages.

– The algorithm SimFrank is constructed as follows. It computes $c \leftarrow \mathsf{Encap}_{\mathsf{c}}^*(\mathsf{pp}; r_{\mathsf{c}}^*)$ and $k_{\mathsf{J}} \leftarrow \mathsf{dSamEncK}(\mathsf{pp}, t; r_{\mathsf{k}}^*)$ with randomness r_{c}^* and r_{k}^*, respectively, where t is a random tag, decapsulates c by Decap using sk_{r} to obtain k_{r}, and then computes c_{t} via encrypting t with the encryption algorithm of USPCE, using the message m as an item. After that, taking $(\perp, t, \perp, r_{\mathsf{c}}^*, r_{\mathsf{k}}^*, r_{\mathsf{USPCE}})$ as the witness, it calls the proving algorithm of NIZK to generate a proof π. Finally, it outputs a signature $\sigma = (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathrm{t}})$.

– Frank computes $(c \leftarrow \mathsf{Encap_c}(\mathsf{pp}; r), k_{\mathsf{r}} \leftarrow \mathsf{Encap_k}(\mathsf{pp}, pk_{\mathsf{r}}; r), k_{\mathsf{J}} \leftarrow \mathsf{dEncap_k}(\mathsf{pp}, pk'_{\mathsf{J}}, t; r))$ with the same randomness r, where t is a random tag, and computes c_{t} via encrypting t with the encryption algorithm of USPCE, using the message m as an item.
Note that for $c \leftarrow \mathsf{Encap_c}(\mathsf{pp}; r)$, we obtain $k_{\mathsf{r}} = \mathsf{Encap_k}(\mathsf{pp}, pk_{\mathsf{r}}; r) = \mathsf{Decap}(\mathsf{pp}, sk_{\mathsf{r}}, c)$ and $k_{\mathsf{J}} = \mathsf{dEncap_k}(\mathsf{pp}, pk'_{\mathsf{J}}, t; r) = \mathsf{dDecap}(\mathsf{pp}, sk'_{\mathsf{J}}, t, c)$. The indistinguishability of dual HPS-KEM$^{\Sigma}$ and the confidentiality against users of USPCE guarantee that the tuple $(c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathrm{t}})$ output by Frank is indistinguishable from $(\widehat{c}, \widehat{k_{\mathsf{r}}}, \widehat{k_{\mathsf{J}}}, \widehat{c_{\mathrm{t}}})$, where $\widehat{c} \leftarrow \mathsf{Encap_c^*}(\mathsf{pp}; r_{\mathsf{c}}^*)$, $\widehat{k_{\mathsf{r}}} = \mathsf{Decap}(\mathsf{pp}, sk_{\mathsf{r}}, \widehat{c})$, $\widehat{k_{\mathsf{J}}} = \mathsf{dDecap}(\mathsf{pp}, sk'_{\mathsf{J}}, t, \widehat{c})$ and $\widehat{c_{\mathrm{t}}}$ is the ciphertext created via encrypting another random tag t' with the encryption algorithm of USPCE, using the message m as an item. Due to the special extended smoothness of dual HPS-KEM$^{\Sigma}$, it guarantees that the tuple $(\widehat{c}, \widehat{k_{\mathsf{r}}}, \widehat{k_{\mathsf{J}}}, \widehat{c_{\mathrm{t}}})$ is indistinguishable from $(\widehat{c}, \widehat{k_{\mathsf{r}}}, \widetilde{k_{\mathsf{J}}}, \widehat{c_{\mathrm{t}}})$, where $\widetilde{k_{\mathsf{J}}}$ is uniformly distributed over the encapsulated key space. According to the uniformity of sampled key by algorithm $\mathsf{dSamEncK}$ of dual HPS-KEM$^{\Sigma}$, $(\widehat{c}, \widehat{k_{\mathsf{r}}}, \widetilde{k_{\mathsf{J}}}, \widehat{c_{\mathrm{t}}})$ is indistinguishable from that $(\widehat{c}, \widehat{k_{\mathsf{r}}}, \overline{k_{\mathsf{J}}}, \widehat{c_{\mathrm{t}}})$, where $\overline{k_{\mathsf{J}}} \leftarrow \mathsf{dSamEncK}(\mathsf{pp}, t; r_{\mathsf{k}}^*)$. The confidentiality against users of USPCE ensures that $(\widehat{c}, \widehat{k_{\mathsf{r}}}, \overline{k_{\mathsf{J}}}, \widehat{c_{\mathrm{t}}})$ is indistinguishable from $(\widehat{c}, \widehat{k_{\mathsf{r}}}, \overline{k_{\mathsf{J}}}, c_{\mathrm{t}})$, which is the tuple output by $\mathsf{SimFrank}$. Thus, the output tuple $(c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathrm{t}})$ from Frank and that from $\mathsf{SimFrank}$ are indistinguishable.

– Frank generates a NIZK proof π for relation $\mathcal{R}$ with witness $(sk_{\mathsf{s}}, t, r, \perp, \perp, r_{\mathsf{USPCE}})$, while $\mathsf{SimFrank}$ generates π for $\mathcal{R}$ with witness $(\perp, t, \perp, r_{\mathsf{c}}^*, r_{\mathsf{k}}^*, r_{\mathsf{USPCE}})$. The zero knowledge property of NIZK guarantees that anyone cannot distinguish the proof output by Frank from that output by $\mathsf{SimFrank}$.

Roughly, confidentiality of sets requires the legislative agency's public key and the judge's public key will not disclose any information about the set of illegal messages (except for its size), which is trivially obtained from the confidentiality of sets of USPCE.

1.2 Discussions

One-Time Token for Specific MAMF Signature. In this paper, our focus is solely on illegal messages. It is worth noting that certain messages, like harassment messages and phishing links, may not universally qualify as illegal messages for all users. Consequently, these messages might not be encompassed within the set designated by the legislative agency. Therefore, without the assistance of the legislative agency, the moderator cannot ascertain the sender's identity in such scenarios. On the other hand, if the legislative agency provides tokens for the messages, the moderator possesses the ability to identify all senders of these messages. To address this, a solution is to empower the legislative agency to generate a one-time token for a specific MAMF signature and a specific message, which is not in pre-defined set, such that the moderator can carry out content moderation for that specific signature and specific message. We stress that our scheme seamlessly accommodates this requirement, leveraging the inherent flexibility of our USPCE. Further elaboration on this aspect can be found in the full version of this paper.

MPC for the Token Generation. In our scheme, the token generation algorithm is invoked by legislative agency, which implicitly means that we assume that the agency would not augment message to illegal set arbitrarily. To mitigate trust in the agency, there exist some general methods. Essentially, the secret key used to generate tokens can be shared among multiple agencies using secret sharing techniques, and then secure multi-party computation (MPC) can be invoked to generate (one-time) tokens for messages deemed illegal by the majority.

Witness-Only Sigma Protocols. When building the Sigma protocol for the aforementioned relation $\mathcal{R}$, partially composed of sub-relations using $\wedge_{\mathrm{eq}}$ operations, we introduce a novel property termed "witness-only" for Sigma protocols, which may have independent interests. Roughly speaking, if the prover in Sigma protocols can generate the commitment and response solely based on the input witness, without necessitating the use of the statement, then we characterize these Sigma protocols as witness-only. It's noteworthy that numerous Sigma protocols inherently possess this witness-only property. Subsequently, we illustrate the construction of a Sigma protocol for a relation composed of sub-relations using an $\wedge_{\mathrm{eq}}$ operation, provided that there exists a witness-only Sigma protocol for each sub-relation. Additional details are available in the full version of this paper.

Predicate-Based Primitive. Similar to the construction of [2], our MAMF is constructed with polynomial-size sets of illegal messages. One might prefer to constructing MAMF with sets of illegal messages expressed with predicates, allowing the scheme to be applied to a broader range of scenarios. We leave it as an open problem to construct a practical MAMF without using cumbersome cryptographic tools (e.g., witness encryption or indistinguishability obfuscation).

2 Preliminaries

Throughout this paper, let λ denote the security parameter. For any $k \in \mathbb{N}$, let $[k] := \{1, 2, \cdots, k\}$. For a finite set S, we denote by $|S|$ the number of elements in S, and denote by $a \leftarrow S$ the process of uniformly sampling a from S. For a distribution X, we denote by $a \leftarrow X$ the process of sampling a from X. For any probabilistic polynomial-time (PPT) algorithm Alg, let $\mathcal{RS}$ be the randomness space of Alg. We write $\mathsf{Alg}(x; r)$ for the process of Alg on input x with inner randomness $r \in \mathcal{RS}$, and use $y \leftarrow \mathsf{Alg}(x)$ to denote the process of running Alg on input x with $r \leftarrow \mathcal{RS}$, and assigning y the result. We write $\mathsf{negl}(\lambda)$ to denote a negligible function in λ and write $\mathsf{poly}(\lambda)$ to denote a polynomial.

For a polynomial-time relation $\mathcal{R} \subset \mathcal{Y} \times \mathcal{X}$, where $\mathcal{Y}$ is the statement space and $\mathcal{X}$ is the witness space, we say that x is a witness for y if $(y, x) \in \mathcal{R}$.

Due to space limitations, please refer the other preliminaries to the full version of this paper, including some cryptographic assumptions, the definitions of NIZK and Sigma protocols, the definition of cuckoo hash and a summary of set pre-constrained encryption.

3 Mild Asymmetric Message Franking

In this section, we introduce a primitive known as *mild asymmetric message franking (MAMF)*, to establish mild content moderation for a messaging system, and formally define its security notions.

3.1 MAMF Algorithms

Formally, an MAMF scheme $\mathsf{MAMF} = (\mathsf{Setup}, \mathsf{KG}_{\mathsf{Ag}}, \mathsf{KG}_{\mathsf{J}}, \mathsf{KG}_{\mathsf{u}}, \mathsf{Frank}, \mathsf{Verify}, \mathsf{TKGen}, \mathsf{Judge}, \mathsf{Forge}, \mathsf{RForge}, \mathsf{JForge})$ is a tuple of algorithms, encompassing four roles: a sender, a receiver, a legislative agency, and a judge. The scheme is associated with three public key spaces (for the legislative agency, the judge, and users, including senders and receivers, respectively), three corresponding secret key spaces, a message space $\mathcal{M}$, a token space $\mathcal{TK}$, and a signature space $\mathcal{SG}$. Without loss of generality, we assume that all public key inputs are in the corresponding public key space, all secret key inputs are in the corresponding secret key space, all m inputs are in $\mathcal{M}$, all tk inputs are in $\mathcal{TK}$, and all σ inputs are in $\mathcal{SG}$.

The detailed descriptions of the algorithms are as follows.

- $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$: The setup algorithm takes the security parameter as input, and outputs a global public parameter pp.
- $(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}}) \leftarrow \mathsf{KG}_{\mathsf{Ag}}(\mathsf{pp}, \mathsf{S})$: The key generation algorithm $\mathsf{KG}_{\mathsf{Ag}}$ takes pp and a set $\mathsf{S} \subseteq \mathcal{M}$ as input, and outputs a key pair $(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}})$ for the legislative agency and an auxiliary parameter $\mathsf{ap}_{\mathsf{Ag}}$ for the judge's key generation.
- $(pk_{\mathsf{J}}, sk_{\mathsf{J}}) \leftarrow \mathsf{KG}_{\mathsf{J}}(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$: The key generation algorithm KG_{J} takes $(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$ as input, and outputs a key pair $(pk_{\mathsf{J}}, sk_{\mathsf{J}})$ for the judge. We assume that the well-formedness of the public key pk_{J} can be verified with the assistance of $\mathsf{ap}_{\mathsf{Ag}}$ and S.
- $(pk, sk) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$: The key generation algorithm KG_{u} takes pp as input, and outputs a key pair (pk, sk) for users. Below we usually use $(pk_{\mathsf{s}}, sk_{\mathsf{s}})$ (resp., $(pk_{\mathsf{r}}, sk_{\mathsf{r}})$) to denote the sender's (resp., the receiver's) public/secret key pair.
- $\sigma \leftarrow \mathsf{Frank}(\mathsf{pp}, sk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m)$: The franking algorithm takes the public parameter pp, a sender's secret key sk_{s}, a receiver's public key pk_{r}, the agency's public key pk_{Ag}, the judge's public key pk_{J} and a message m as input, and outputs a signature σ.
- $b \leftarrow \mathsf{Verify}(\mathsf{pp}, pk_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m, \sigma)$: The *deterministic* algorithm of the verification of the receiver takes $(\mathsf{pp}, pk_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}})$, a message m and a signature σ as input, and returns $b \in \{0, 1\}$, which indicates whether the receiver accepts the signature or not.

- $\mathsf{tk} \leftarrow \mathsf{TKGen}(\mathsf{pp}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m)$: The token generation algorithm, run by the agency, takes $(\mathsf{pp}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}})$ and a message m as input, and outputs a token tk.
- $b \leftarrow \mathsf{Judge}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m, \sigma, \mathsf{tk})$: The *deterministic* algorithm of verification of the judge takes $(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}})$, a message m, a signature σ and a token tk as input, and outputs a bit $b \in \{0, 1\}$. Note that, when $m \in \mathsf{S}$, the token tk can be $\perp$.
- $\sigma \leftarrow \mathsf{Forge}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m)$: The universal forging algorithm, on input $(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}})$ and a message m, returns a "forged" signature σ.
- $\sigma \leftarrow \mathsf{RForge}(\mathsf{pp}, pk_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m)$: The receiver compromise forging algorithm takes $(\mathsf{pp}, pk_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}})$ and a message m as input, and returns a "forged" signature σ.
- $\sigma \leftarrow \mathsf{JForge}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m)$: The judge compromise forging algorithm takes $(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}})$ and a message m as input, and outputs a "forged" signature σ.

Correctness. For any normal signature generated by Frank, the correctness requires that (i) the receiver can call Verify to verify the signature successfully, and (ii) the judge can invoke Judge to validate a report successfully once they receive a valid report. The formal requirements are shown as follows.

Given any pp generated by Setup, any key pairs $(pk_{\mathsf{s}}, sk_{\mathsf{s}})$ and $(pk_{\mathsf{r}}, sk_{\mathsf{r}})$ output by KG_{u}, any key pair $(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$ for a set $\mathsf{S} \subseteq \mathcal{M}$ output by $\mathsf{KG}_{\mathsf{Ag}}$ (where S is selected by some authority, e.g., the agency), and any key pair $(pk_{\mathsf{J}}, sk_{\mathsf{J}})$ output by KG_{J} (with $(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$ as input), we require that for any message $m \in \mathcal{M}$ and any $\sigma \leftarrow \mathsf{Frank}(\mathsf{pp}, sk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m)$, it holds with overwhelming probability that:

(1) $\mathsf{Verify}(\mathsf{pp}, pk_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m, \sigma) = 1$;
(2) if $m \in \mathsf{S}$, then $\mathsf{Judge}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m, \sigma, \mathsf{tk} = \perp) = 1$;
(3) if $m \notin \mathsf{S}$, then $\mathsf{tk} \leftarrow \mathsf{TKGen}(\mathsf{pp}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m)$, and $\mathsf{Judge}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m, \sigma, \mathsf{tk}) = 1$.

Remark 1. For the sake of simplicity, we posit the existence of an additional key verification algorithms $\mathsf{WellForm}_{\mathsf{u}}$ and a token verification algorithm $\mathsf{WellForm}_{\mathsf{tk}}$. The algorithm $\mathsf{WellForm}_{\mathsf{u}}$ takes the public parameter and the key pairs of the receiver (or sender) as input, producing a bit b that signifies the well-formedness of the key pairs. The algorithm $\mathsf{WellForm}_{\mathsf{tk}}$ takes the public parameter, the judge's public key and the token as input, producing a bit b that signifies the well-formedness of the token.

3.2 Security Notions for MAMF

We now formalize specific security notions for MAMF, including unforgeability, accountability, deniability, unframeability, untraceability, and confidentiality of sets.

Unforgeability. One of the paramount security requirements in secure messaging applications is the prevention of malicious impersonation. In essence, MAMF must guarantee that successful impersonation is thwarted, contingent upon the non-compromise of the respective individual's secret key.

Definition 1 (Unforgeability). *An MAMF scheme* MAMF *is* unforgeable*, if for any set* $\mathsf{S} \subseteq \mathcal{M}$ *and any PPT adversary* $\mathcal{A}$*,* $\mathbf{Adv}^{\text{unforge}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) := \Pr[\mathbf{G}^{\text{unforge}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{G}^{\text{unforge}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$ *is shown in Fig. 2.*

Accountability. Adhering to the terminology established in AMF [18], we systematically formalize the security requirement pertaining to accountability as *receiver binding* and *sender binding*. Concretely, MAMF is to ensure that (i) no receivers can deceive the judge into accepting a message not genuinely sent by the sender, and (ii) no sender can produce a signature acceptable to the receiver while simultaneously rejected by the judge.

Now, we present the formal definitions as below.

Definition 2 (r-BIND). *An MAMF scheme* MAMF *is* receiver-binding*, if for any set* $\mathsf{S} \subseteq \mathcal{M}$ *and any PPT adversary* $\mathcal{A}$*,* $\mathbf{Adv}^{\text{r-bind}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) := \Pr[\mathbf{G}^{\text{r-bind}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{G}^{\text{r-bind}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$ *is shown in Fig. 2.*

Definition 3 (s-BIND). *An MAMF scheme* MAMF *is* sender-binding*, if for any set* $\mathsf{S} \subseteq \mathcal{M}$ *and any PPT adversary* $\mathcal{A}$*,* $\mathbf{Adv}^{\text{s-bind}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) := \Pr[\mathbf{G}^{\text{s-bind}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{G}^{\text{s-bind}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$ *is shown in Fig. 2.*

Deniability. To uphold deniability, MAMF needs to adhere to the secure properties of *universal deniability*, *receiver compromise deniability*, and *judge compromise deniability*.

Universal deniability indicates that any non-participating entity (i.e., lacking access to the sender's secret key, the receiver's secret key, or the judge's secret key) can generate a signature, indistinguishable from honestly-created signatures to other non-participating entities.

For receiver compromise deniability, the property requires that an entity with access to the receiver's secret key can generate a signature. This generated signature should be indistinguishable from honestly-created signatures to other entities with access to the corrupted secret key of the receiver.

As for judge compromise deniability, an entity with access to the judge's secret key should be capable of creating a signature. This signature should be indistinguishable from honestly-generated signatures for other entities with access to the judge's secret key.

The formal definitions are presented as follows.

Definition 4 (UnivDen). *An MAMF scheme* MAMF *is* universally deniable*, if for any set* $\mathsf{S} \subseteq \mathcal{M}$ *and any PPT adversary* $\mathcal{A}$*,* $\mathbf{Adv}^{\text{UnivDen}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) := |\Pr[\mathbf{G}^{\text{UnivDen}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{G}^{\text{UnivDen}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$ *is shown in Fig. 2.*

```
$\mathbf{G}^{\mathrm{unforge}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$:
    $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$, $Q_{\mathrm{sig}} := \emptyset$
    $(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}}) \leftarrow \mathsf{KG}_{\mathsf{Ag}}(\mathsf{pp}, \mathsf{S})$
    $(pk_{\mathsf{J}}, sk_{\mathsf{J}}) \leftarrow \mathsf{KG}_{\mathsf{J}}(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$
    $(pk_{\mathsf{s}}, sk_{\mathsf{s}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$, $(pk_{\mathsf{r}}, sk_{\mathsf{r}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$
    $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}}, sk_{\mathsf{J}})$
    If $(pk_{\mathsf{r}}, m^*) \in Q_{\mathrm{sig}}$: Return 0
    Return $\mathsf{Verify}(\mathsf{pp}, pk_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m^*, \sigma^*)$

$\mathcal{O}^{\mathsf{Frank}}(pk'_{\mathsf{r}}, m')$:
    $\sigma' \leftarrow \mathsf{Frank}(\mathsf{pp}, sk_{\mathsf{s}}, pk'_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m')$
    $Q_{\mathrm{sig}} \leftarrow Q_{\mathrm{sig}} \cup \{(pk'_{\mathsf{r}}, m')\}$
    Return $\sigma'$

$\mathcal{O}^{\mathsf{Verify}}(pk'_{\mathsf{s}}, m', \sigma')$:
    Return $\mathsf{Verify}(\mathsf{pp}, pk'_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m', \sigma')$
```

```
$\mathbf{G}^{\mathrm{r\text{-}bind}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$:
    $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$, $Q_{\mathrm{sig}} := \emptyset$
    $(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}}) \leftarrow \mathsf{KG}_{\mathsf{Ag}}(\mathsf{pp}, \mathsf{S})$
    $(pk_{\mathsf{J}}, sk_{\mathsf{J}}) \leftarrow \mathsf{KG}_{\mathsf{J}}(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$, $(pk_{\mathsf{s}}, sk_{\mathsf{s}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$
    $(pk^*_{\mathsf{r}}, m^*, \sigma^*, \mathsf{tk}^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, pk_{\mathsf{s}}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}})$
    If $(pk^*_{\mathsf{r}}, m^*) \in Q_{\mathrm{sig}}$: Return 0
    Return $\mathsf{Judge}(\mathsf{pp}, pk_{\mathsf{s}}, pk^*_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m^*, \sigma^*, \mathsf{tk}^*)$

$\mathcal{O}^{\mathsf{Frank}}(pk'_{\mathsf{r}}, m')$:
    $\sigma' \leftarrow \mathsf{Frank}(\mathsf{pp}, sk_{\mathsf{s}}, pk'_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m')$
    $Q_{\mathrm{sig}} \leftarrow Q_{\mathrm{sig}} \cup \{(pk'_{\mathsf{r}}, m')\}$
    Return $\sigma'$

$\mathcal{O}^{\mathsf{Judge}}(pk'_{\mathsf{s}}, pk'_{\mathsf{r}}, m', \sigma', \mathsf{tk}')$:
    Return $\mathsf{Judge}(\mathsf{pp}, pk'_{\mathsf{s}}, pk'_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m', \sigma', \mathsf{tk}')$
```

```
$\mathbf{G}^{\mathrm{s\text{-}bind}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$:
    $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$, $(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}}) \leftarrow \mathsf{KG}_{\mathsf{Ag}}(\mathsf{pp}, \mathsf{S})$
    $(pk_{\mathsf{J}}, sk_{\mathsf{J}}) \leftarrow \mathsf{KG}_{\mathsf{J}}(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$, $(pk_{\mathsf{r}}, sk_{\mathsf{r}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$
    $(pk^*_{\mathsf{s}}, m^*, \sigma^*, \mathsf{tk}^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, pk_{\mathsf{r}}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}})$
    If $(m^* \notin \mathsf{S}) \wedge (\mathsf{WellForm}_{\mathsf{tk}}(\mathsf{pp}, pk_{\mathsf{J}}, \mathsf{tk}^*) = 0)$: Return 0
    If $(m^* \in \mathsf{S}) \wedge (\mathsf{tk}^* \neq \perp)$: Return 0
    $b_1 \leftarrow \mathsf{Verify}(\mathsf{pp}, pk^*_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m^*, \sigma^*)$
    $b_2 \leftarrow \mathsf{Judge}(\mathsf{pp}, pk^*_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m^*, \sigma^*, \mathsf{tk}^*)$
    Return $b_1 \wedge \neg b_2$

$\mathcal{O}^{\mathsf{Verify}}(pk'_{\mathsf{s}}, m', \sigma')$:
    Return $\mathsf{Verify}(\mathsf{pp}, pk'_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m', \sigma')$

$\mathcal{O}^{\mathsf{Judge}}(pk'_{\mathsf{s}}, pk'_{\mathsf{r}}, m', \sigma', \mathsf{tk}')$:
    Return $\mathsf{Judge}(\mathsf{pp}, pk'_{\mathsf{s}}, pk'_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m', \sigma', \mathsf{tk}')$
```

```
$\mathbf{G}^{\mathrm{UnivDen}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$:
    $b \leftarrow \{0,1\}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$, $(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}}) \leftarrow \mathsf{KG}_{\mathsf{Ag}}(\mathsf{pp}, \mathsf{S})$
    $(pk_{\mathsf{J}}, sk_{\mathsf{J}}) \leftarrow \mathsf{KG}_{\mathsf{J}}(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$
    $(pk_{\mathsf{s}}, sk_{\mathsf{s}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$, $(pk_{\mathsf{r}}, sk_{\mathsf{r}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$
    $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, sk_{\mathsf{s}}, pk_{\mathsf{r}}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}})$
    Return $(b' = b)$

$\mathcal{O}^{\mathsf{F\text{-}F}}(m')$:
    $\sigma_0 \leftarrow \mathsf{Frank}(\mathsf{pp}, sk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m')$
    $\sigma_1 \leftarrow \mathsf{Forge}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m')$
    Return $\sigma_b$
```

```
$\mathbf{G}^{\mathrm{ReComDen}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$:
    $b \leftarrow \{0,1\}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$, $(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}}) \leftarrow \mathsf{KG}_{\mathsf{Ag}}(\mathsf{pp}, \mathsf{S})$
    $(pk_{\mathsf{J}}, sk_{\mathsf{J}}) \leftarrow \mathsf{KG}_{\mathsf{J}}(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$, $(pk_{\mathsf{s}}, sk_{\mathsf{s}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$
    $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, sk_{\mathsf{s}}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}})$
    Return $(b' = b)$

$\mathcal{O}^{\mathsf{F\text{-}RF}}(pk'_{\mathsf{r}}, sk'_{\mathsf{r}}, m')$:
    If $\mathsf{WellForm}_{\mathsf{u}}(\mathsf{pp}, pk'_{\mathsf{r}}, sk'_{\mathsf{r}}) = 0$: Return $\perp$
    $\sigma_0 \leftarrow \mathsf{Frank}(\mathsf{pp}, sk_{\mathsf{s}}, pk'_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m')$
    $\sigma_1 \leftarrow \mathsf{RForge}(\mathsf{pp}, pk_{\mathsf{s}}, sk'_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m')$
    Return $\sigma_b$
```

```
$\mathbf{G}^{\mathrm{JuComDen}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$:
    $b \leftarrow \{0,1\}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$, $(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}}) \leftarrow \mathsf{KG}_{\mathsf{Ag}}(\mathsf{pp}, \mathsf{S})$
    $(pk_{\mathsf{J}}, sk_{\mathsf{J}}) \leftarrow \mathsf{KG}_{\mathsf{J}}(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$
    $(pk_{\mathsf{s}}, sk_{\mathsf{s}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$, $(pk_{\mathsf{r}}, sk_{\mathsf{r}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$
    $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, sk_{\mathsf{s}}, pk_{\mathsf{r}}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}}, sk_{\mathsf{J}})$
    Return $(b' = b)$

$\mathcal{O}^{\mathsf{F\text{-}JF}}(m')$:
    $\sigma_0 \leftarrow \mathsf{Frank}(\mathsf{pp}, sk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m')$
    $\sigma_1 \leftarrow \mathsf{JForge}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m')$
    Return $\sigma_b$
```

Fig. 2. Games for defining unforgeability, accountability and deniability of MAMF

Definition 5 (ReComDen). *An MAMF scheme* MAMF *is* receiver-compromise deniable, *if for any set* $\mathsf{S} \subseteq \mathcal{M}$ *and any PPT adversary* $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{ReComDen}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) := |\Pr[\mathbf{G}^{\mathrm{ReComDen}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$, *where* $\mathbf{G}^{\mathrm{ReComDen}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$ *is shown in Fig. 2.*

Definition 6 (JuComDen). *An MAMF scheme* MAMF *is* judge-compromise deniable, *if for any set* $\mathsf{S} \subseteq \mathcal{M}$ *and any PPT adversary* $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{JuComDen}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) := |\Pr[\mathbf{G}^{\mathrm{JuComDen}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$, *where* $\mathbf{G}^{\mathrm{JuComDen}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$ *is shown in Fig. 2.*

Unframeability. The unframeability of MAMF requires that no party, even given a receiver's secret key and the judge's secret key, is able to produce a signature acceptable to both the receiver and the judge.

The formal definition is as follows.

Definition 7 (Unframeability). *An MAMF scheme* MAMF *is* unframeable*, if for any set* $\mathsf{S} \subseteq \mathcal{M}$ *and any PPT adversary* $\mathcal{A}$*,* $\mathbf{Adv}^{\text{Unframe}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) := |\Pr[\mathbf{G}^{\text{Unframe}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{G}^{\text{Unframe}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$ *is shown in Fig. 3.*

Untraceability. Ensuring untraceability constrains the capabilities of both the agency and the judge, thereby enhancing the assurance of sender privacy. Informally, untraceability implies an inability to discern the exact sender of a message. This concept is formalized into two distinct notions: untraceability against judge and untraceability against agency.

Untraceability Against Judge. When the message is not within the set S, untraceability against judge ensures that an entity possessing the judge's secret key cannot identify the sender of the message, without any assistance from the agency.

Definition 8 (Untraceability against judge). *An MAMF scheme* MAMF *has* untraceability against judge*, if for any set* $\mathsf{S} \subseteq \mathcal{M}$ *and any PPT adversary* $\mathcal{A}$*, there is a simulator* $\mathsf{SimFrank}$*, such that* $\mathbf{Adv}^{\text{Unt-J}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) := |\Pr[\mathbf{G}^{\text{Unt-J}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{G}^{\text{Unt-J}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$ *is shown in Fig. 3.*

Untraceability Against Agency. We also articulated untraceability against agency. In essence, a party with access to the agency's secret key is unable to discern the sender of a given message. The formal definition is articulated as follows.

Definition 9 (Untraceability against agency). *An MAMF scheme* MAMF *has* untraceability against agency*, if for any set* $\mathsf{S} \subseteq \mathcal{M}$ *and any PPT adversary* $\mathcal{A}$*, there is a simulator* $\mathsf{SimFrank}$*, such that* $\mathbf{Adv}^{\text{Unt-Ag}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) := |\Pr[\mathbf{G}^{\text{Unt-Ag}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{G}^{\text{Unt-Ag}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)$ *is shown in Fig. 3.*

Confidentiality of Sets. We also consider the confidentiality of sets. It means the public parameters and the public keys will not disclose any information about the pre-defined set S. The formal definition is outlined as follows.

Definition 10 (Confidentiality of sets). *An MAMF scheme* MAMF *supports* confidentiality of sets*, if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,* $\mathbf{Adv}^{\text{conf-set}}_{\mathsf{MAMF},\mathcal{A}}(\lambda) := |\Pr[\mathbf{G}^{\text{conf-set}}_{\mathsf{MAMF},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{G}^{\text{conf-set}}_{\mathsf{MAMF},\mathcal{A}}(\lambda)$ *is shown in Fig. 3.*

Remark 2. Unlike AMF [18], where receiver binding and sender binding imply unforgeability, our notion differs. The unforgeability in [18] provides an adversary access to a judge oracle, while ours directly provides the judge's secret key. The unforgeability in [18] cannot prevent the receiver from accepting a signature forged by the judge (as discussed in [18, Appendix B], where signatures output by

```
$\underline{\mathbf{G}^{\text{Unframe}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)}$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$, $Q_{\text{sig}} := \emptyset$, $(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}}) \leftarrow \mathsf{KG}_{\mathsf{Ag}}(\mathsf{pp}, \mathsf{S})$
$(pk_{\mathsf{J}}, sk_{\mathsf{J}}) \leftarrow \mathsf{KG}_{\mathsf{J}}(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$
$(pk_{\mathsf{s}}, sk_{\mathsf{s}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$, $(pk_{\mathsf{r}}, sk_{\mathsf{r}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, pk_{\mathsf{s}}, sk_{\mathsf{r}}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}}, sk_{\mathsf{J}})$
If $m^* \in Q_{\text{sig}}$: Return 0
$\mathsf{tk}^* \leftarrow \mathsf{TKGen}(\mathsf{pp}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m^*)$
$b_1 \leftarrow \mathsf{Verify}(\mathsf{pp}, pk_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m^*, \sigma^*)$
$b_2 \leftarrow \mathsf{Judge}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m^*, \sigma^*, \mathsf{tk}^*)$
Return $b_1 \wedge b_2$

$\underline{\mathcal{O}^{\mathsf{Frank}}(m')}$:
$\sigma' \leftarrow \mathsf{Frank}(\mathsf{pp}, sk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m')$
$Q_{\text{sig}} \leftarrow Q_{\text{sig}} \cup \{m'\}$
Return $\sigma'$
```

```
$\underline{\mathbf{G}^{\text{Unt-J}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)}$:
$b \leftarrow \{0,1\}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$, $Q_{\text{m}} := \emptyset$
$(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}}) \leftarrow \mathsf{KG}_{\mathsf{Ag}}(\mathsf{pp}, \mathsf{S})$
$(pk_{\mathsf{J}}, sk_{\mathsf{J}}) \leftarrow \mathsf{KG}_{\mathsf{J}}(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$, $(pk_{\mathsf{s}}, sk_{\mathsf{s}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, sk_{\mathsf{J}})$
Return $(b' = b)$

$\underline{\mathcal{O}^{\mathsf{TKGen}}(m')}$:
If $m' \in Q_{\text{m}}$: Return $\perp$
$\mathsf{tk} \leftarrow \mathsf{TKGen}(\mathsf{pp}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m')$, $Q_{\text{m}} \leftarrow Q_{\text{m}} \cup \{m'\}$
Return $\mathsf{tk}$

$\underline{\mathcal{O}^{\mathsf{Ch}}(pk'_{\mathsf{r}}, sk'_{\mathsf{r}}, m')}$:
If $\mathsf{WellForm}_{\mathsf{u}}(\mathsf{pp}, pk'_{\mathsf{r}}, sk'_{\mathsf{r}}) = 0$: Return $\perp$
If $m' \in \mathsf{S}$ : Return $\perp$
If $m' \in Q_{\text{m}}$: Return $\perp$
$Q_{\text{m}} \leftarrow Q_{\text{m}} \cup \{m'\}$
$\sigma_0 \leftarrow \mathsf{Frank}(\mathsf{pp}, sk_{\mathsf{s}}, pk'_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m')$
$\sigma_1 \leftarrow \mathsf{SimFrank}(\mathsf{pp}, pk_{\mathsf{s}}, sk'_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m')$
Return $\sigma_b$
```

```
$\underline{\mathbf{G}^{\text{Unt-Ag}}_{\mathsf{MAMF},\mathsf{S},\mathcal{A}}(\lambda)}$:
$b \leftarrow \{0,1\}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$, $Q_{\text{ch}} := \emptyset$
$(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}}) \leftarrow \mathsf{KG}_{\mathsf{Ag}}(\mathsf{pp}, \mathsf{S})$
$(pk_{\mathsf{J}}, sk_{\mathsf{J}}) \leftarrow \mathsf{KG}_{\mathsf{J}}(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$, $(pk_{\mathsf{s}}, sk_{\mathsf{s}}) \leftarrow \mathsf{KG}_{\mathsf{u}}(\mathsf{pp})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, pk_{\mathsf{s}}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}})$
Return $(b' = b)$

$\underline{\mathcal{O}^{\mathsf{Judge}}(pk'_{\mathsf{r}}, m', \sigma', \mathsf{tk}')}$:
If $(pk'_{\mathsf{r}}, m') \in Q_{\text{ch}}$: Return $\perp$
Return $\mathsf{Judge}(\mathsf{pp}, pk_{\mathsf{s}}, pk'_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m', \sigma', \mathsf{tk}')$

$\underline{\mathcal{O}^{\mathsf{Ch}}(pk'_{\mathsf{r}}, sk'_{\mathsf{r}}, m')}$:
If $\mathsf{WellForm}_{\mathsf{u}}(\mathsf{pp}, pk'_{\mathsf{r}}, sk'_{\mathsf{r}}) = 0$: Return $\perp$
$\sigma_0 \leftarrow \mathsf{Frank}(\mathsf{pp}, sk_{\mathsf{s}}, pk'_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m')$
$\sigma_1 \leftarrow \mathsf{SimFrank}(\mathsf{pp}, pk_{\mathsf{s}}, sk'_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m')$
$Q_{\text{ch}} \leftarrow Q_{\text{ch}} \cup \{(pk'_{\mathsf{r}}, m')\}$
Return $\sigma_b$
```

```
$\underline{\mathbf{G}^{\text{conf-set}}_{\mathsf{MAMF},\mathcal{A}}(\lambda)}$:
$b \leftarrow \{0,1\}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(\lambda)$
$(\mathsf{S}_0, \mathsf{S}_1, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(\mathsf{pp})$ s.t $(\mathsf{S}_0 \subset \mathcal{M}) \wedge (\mathsf{S}_1 \subset \mathcal{M}) \wedge (|\mathsf{S}_0| = |\mathsf{S}_1|)$
$(pk_{\mathsf{Ag}}, sk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}}) \leftarrow \mathsf{KG}_{\mathsf{Ag}}(\mathsf{pp}, \mathsf{S}_b)$, $(pk_{\mathsf{J}}, sk_{\mathsf{J}}) \leftarrow \mathsf{KG}_{\mathsf{J}}(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})$
$b' \leftarrow \mathcal{A}_2(\mathsf{pp}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, st_{\mathcal{A}})$
Return $(b' = b)$
```

Fig. 3. Games for defining unframeability, untraceability, and confidentiality of sets of MAMF

its JForge algorithm can be accepted by the receiver). Our unforgeability ensures that the receiver will not accept a signature forged by anyone else, including the agency and the judge, thus preventing deception.

In [18], the adversary in the judge compromise deniability game can access to *all* secret keys (including the sender's, the receiver's and the judge's). It means that the signature output by JForge can be accept by the receiver, which is unreasonable. In our model, the adversary only has access to the sender's and the judge's secret keys but *not to the receiver's secret key*. More importantly, the definition of judge compromise deniability in [18] is contradictory to unframeability. Our deniability model makes space for unframeability.

Our notions have some areas to be strengthened, e.g., strong unforgeability. We believe the current definitions have grasped the key security requirements of MAMF. Some enhancement definitions can be considered for future work.

4 Universal Set Pre-constrained Encryption

In this section, we introduce a new primitive, *universal set pre-constrained encryption (USPCE).*

Definition. Let $\mathcal{U}$ denote a universe of elements, and $\mathcal{M}$ denote a message space. The universal set pre-constrained encryption USPCE contains five algorithms $(\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{TKGen}, \mathsf{Dec})$ and the details are as follows.

- $(\mathsf{pp}, \mathsf{ap}, msk) \leftarrow \mathsf{Setup}(\lambda, \mathsf{S})$: The setup algorithm, run by the authority, takes as input a security parameter λ and a set $\mathsf{S} \subseteq \mathcal{U}$ of size at most n, and outputs a public parameter pp, a auxiliary parameter ap and a master secret key msk.
- $(pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp}, \mathsf{ap})$: The key generation algorithm is run by the users. It takes $(\mathsf{pp}, \mathsf{ap})$ as input, and outputs a public key pk and a secret key sk. We assume that the well-formedness of pk can be verified with the assistance of ap and S.
- $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, pk, x, m)$: The encryption algorithm takes (pp, pk), an item $x \in \mathcal{U}$ and a message $m \in \mathcal{M}$ as input, and outputs a ciphertext ct.
- $\mathsf{tk} \leftarrow \mathsf{TKGen}(\mathsf{pp}, msk, x)$: The token generation algorithm takes (pp, msk, x) as input, and outputs a token tk for x.
- $m/S_{\mathrm{m}} \leftarrow \mathsf{Dec}(\mathsf{pp}, sk, \mathsf{ct}, \mathsf{tk})$: The decryption algorithm takes $(\mathsf{pp}, sk, \mathsf{ct}, \mathsf{tk})$ as input, and outputs either a message m or a polynomial-size set $S_{\mathrm{m}} \subset \mathcal{M}$.

Given a set $\mathsf{S} \subseteq \mathcal{U}$, for any pp and msk generated by $\mathsf{Setup}(\lambda, \mathsf{S})$, we define a relation as follows:

$$\mathcal{R}_{\mathsf{ct}} = \{((pk, x, \mathsf{ct}), (m, r)) : \mathsf{ct} = \mathsf{Enc}(\mathsf{pp}, pk, x, m; r)\}. \tag{1}$$

We require that there is a *witness-only* Sigma protocol (please refer to the definition of witness-only in the full version of this paper) for the relation $\mathcal{R}_{\mathsf{ct}}$ in Eq. (1).

It also satisfies the following properties.

Definition 11 (Correctness). *A USPCE scheme* USPCE *is* correct*, if for any* $\lambda \in \mathbb{N}$*, any set* $\mathsf{S} \subset \mathcal{U}$*, and any* $m \in \mathcal{M}$*, it holds that*

– *when* $x \in \mathsf{S}$*:*

$$\Pr\left[\begin{array}{l}(\mathsf{pp}, \mathsf{ap}, msk) \leftarrow \mathsf{Setup}(\lambda, \mathsf{S}) \\ (pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp}, \mathsf{ap}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, pk, x, m)\end{array} : m \in S_{\mathrm{m}} = \mathsf{Dec}(\mathsf{pp}, sk, \mathsf{ct}, \perp)\right] = 1 - \mathsf{negl}(\lambda);$$

– *when* $x \notin \mathsf{S}$*:*

$$\Pr\left[\begin{array}{l}(\mathsf{pp}, \mathsf{ap}, msk) \leftarrow \mathsf{Setup}(\lambda, \mathsf{S}) \\ (pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp}, \mathsf{ap}) \\ \mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, pk, x, m) \\ \mathsf{tk} \leftarrow \mathsf{TKGen}(\mathsf{pp}, msk, x)\end{array} : m = \mathsf{Dec}(\mathsf{pp}, sk, \mathsf{ct}, \mathsf{tk})\right] = 1 - \mathsf{negl}(\lambda).$$

Confidentiality Against Authority. Here, we address the matter of confidentiality against the authority. In a nutshell, the authority cannot obtain meaningful information about the message from a ciphertext.

```
$\mathbf{G}^{\text{conf-au}}_{\mathsf{USPCE},\mathcal{A},\mathsf{S}}(\lambda)$:
$b \leftarrow \{0,1\}$, $(\mathsf{pp}, \mathsf{ap}, msk) \leftarrow \mathsf{Setup}(\lambda, \mathsf{S})$, $(pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp}, \mathsf{ap})$
$(m_0, m_1, x^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(\mathsf{pp}, msk, pk)$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, pk, x^*, m_b)$, $b' \leftarrow \mathcal{A}_2(\mathsf{ct}, st_{\mathcal{A}})$
Return $(b' = b)$
```

```
$\mathbf{G}^{\text{conf-u}}_{\mathsf{USPCE},\mathcal{A},\mathsf{S}}(\lambda)$:
$b \leftarrow \{0,1\}$, $(\mathsf{pp}, \mathsf{ap}, msk) \leftarrow \mathsf{Setup}(\lambda, \mathsf{S})$, $Q_{\mathrm{x}} := \emptyset$, $U_{\mathrm{x}} := \emptyset$
$(pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp}, \mathsf{ap})$, $(m_0, m_1, x^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1^{\mathcal{O}}(\mathsf{pp}, pk, sk)$
If $(x^* \notin \mathcal{U}) \vee (x^* \in \mathsf{S}) \vee (x^* \in Q_{\mathrm{x}})$: Return $\perp$
$U_{\mathrm{x}} \leftarrow U_{\mathrm{x}} \cup \{x^*\}$, $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, pk, x^*, m_b)$, $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(\mathsf{ct}, st_{\mathcal{A}})$
Return $(b' = b)$

$\mathcal{O}^{\mathsf{TKGen}}(x')$:
If $x' \in U_{\mathrm{x}}$: Return $\perp$
$Q_{\mathrm{x}} \leftarrow Q_{\mathrm{x}} \cup \{x'\}$
Return $\mathsf{TKGen}(\mathsf{pp}, msk, x')$
```

```
$\mathbf{G}^{\text{conf-set}}_{\mathsf{USPCE},\mathcal{A}}(\lambda)$:
$b \leftarrow \{0,1\}$, $(\mathsf{S}_0, \mathsf{S}_1, st_{\mathcal{A}}) \leftarrow \mathcal{A}_1(\lambda)$ s.t $(\mathsf{S}_0 \subset \mathcal{U}) \wedge (\mathsf{S}_1 \subset \mathcal{U}) \wedge (|\mathsf{S}_0| = |\mathsf{S}_1|)$
$(\mathsf{pp}, \mathsf{ap}, msk) \leftarrow \mathsf{Setup}(\lambda, \mathsf{S}_b)$, $(pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp}, \mathsf{ap})$, $b' \leftarrow \mathcal{A}_2(\mathsf{pp}, pk, st_{\mathcal{A}})$
Return $(b' = b)$
```

Fig. 4. Games $\mathbf{G}^{\text{conf-au}}_{\mathsf{USPCE},\mathcal{A},\mathsf{S}}(\lambda)$, $\mathbf{G}^{\text{conf-u}}_{\mathsf{USPCE},\mathcal{A},\mathsf{S}}(\lambda)$ and $\mathbf{G}^{\text{conf-set}}_{\mathsf{USPCE},\mathcal{A}}(\lambda)$ for USPCE

Definition 12 (Confidentiality against authority). *A USPCE scheme* USPCE *has* confidentiality against authority*, if for any set* $\mathsf{S} \subseteq \mathcal{U}$ *and any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,* $\mathbf{Adv}^{\text{conf-au}}_{\mathsf{USPCE},\mathcal{A},\mathsf{S}}(\lambda) := |\Pr[\mathbf{G}^{\text{conf-au}}_{\mathsf{USPCE},\mathcal{A},\mathsf{S}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{G}^{\text{conf-au}}_{\mathsf{USPCE},\mathcal{A},\mathsf{S}}(\lambda)$ *is shown in Fig. 4.*

Confidentiality Against Users. We extend our considerations to confidentiality against users. Informally, It is required that, without the token for an item $x \notin \mathsf{S}$ given by the authority, any user cannot obtain meaningful information about the message from a ciphertext associated with x.

Definition 13 (Confidentiality against users). *A USPCE scheme* USPCE *has* confidentiality against users*, if for any set* $\mathsf{S} \subseteq \mathcal{U}$ *and any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,* $\mathbf{Adv}^{\text{conf-u}}_{\mathsf{USPCE},\mathcal{A},\mathsf{S}}(\lambda) := |\Pr[\mathbf{G}^{\text{conf-u}}_{\mathsf{USPCE},\mathcal{A},\mathsf{S}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{G}^{\text{conf-u}}_{\mathsf{USPCE},\mathcal{A},\mathsf{S}}(\lambda)$ *is shown in Fig. 4.*

Confidentiality of Sets. Then, we delve into the concept of confidentiality of sets. It is required that the public parameters and a user's public key will not disclose any information about the pre-defined set S.

Definition 14 (Confidentiality of sets). *A USPCE scheme* USPCE *supports* confidentiality of sets*, if for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*,* $\mathbf{Adv}^{\text{conf-set}}_{\mathsf{USPCE},\mathcal{A}}(\lambda) := |\Pr[\mathbf{G}^{\text{conf-set}}_{\mathsf{USPCE},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$*, where* $\mathbf{G}^{\text{conf-set}}_{\mathsf{USPCE},\mathcal{A}}(\lambda)$ *is shown in Fig. 4.*

Construction. Let $\mathsf{CH}^{(\mathsf{rob})}_{\lambda} = (\mathsf{Setup}, \mathsf{Insert}, \mathsf{Lookup})$ be an ϵ-robust cuckoo hashing scheme (which is outlined in the full version of this paper), of which the negligibility of ϵ is ensured. More exactly, given the security parameter λ, the setup algorithm chooses $k = \lambda$ hash functions, the size of the hash table is $n' = 2 \cdot \lambda \cdot n$, and the size of the stash is 0. Let GenG be a group generation algorithm for bilinear maps, which takes the security parameter as input and outputs the group

```
$\underline{\mathsf{Setup}(\lambda, \mathsf{S}):}$
$(e, \mathbb{G}, \mathbb{G}_T, g, p) \leftarrow \mathsf{GenG}(\lambda)$    $/\!/ g$ is the generator of $\mathbb{G}$ with order $p$.
Choose hash functions $\widetilde{\mathsf{H}}, \overline{\mathsf{H}} : \mathcal{U} \to \mathbb{G}^*$.
$(\mathsf{pp}_{\mathsf{CH}}, T_{\mathsf{init}}, ST) \leftarrow \mathsf{CH}_{\lambda}^{(\mathsf{rob})}.\mathsf{Setup}(\lambda, n)$    $/\!/ n = \mathsf{poly}(\lambda)$ and $|T_{\mathsf{init}}| = n' = \mathsf{poly}(n) = \mathsf{poly}(\lambda)$.
    $/\!/ \mathsf{pp}_{\mathsf{CH}}$ contains $k$ random hash functions $(\mathsf{H}_j : \mathcal{U} \to [n'])_{j \in [k]}$, $T_{\mathsf{init}}$ is the hash table, $ST$ is the stash.
$(T_{\mathsf{S}}, ST) \leftarrow \mathsf{CH}_{\lambda}^{(\mathsf{rob})}.\mathsf{Insert}(pp_{\mathsf{CH}}, T_{\mathsf{init}}, ST, \mathsf{S})$, $\alpha' \leftarrow \mathbb{Z}_p^*$, $A' := g^{\alpha'}$, $s \leftarrow \mathbb{Z}_p^*$, $Y' := g^s$
Initialize two empty tables $\widetilde{T}, T'$ with length $n'$.
For each $i \in [n']$:
    If $T_{\mathsf{S}}[i] = \perp$:    $\widetilde{T}[i] \leftarrow \mathbb{G}$, $T'[i] := (\widetilde{T}[i])^{\alpha'}$
    Else    $\widetilde{T}[i] := \widetilde{\mathsf{H}}(T_{\mathsf{S}}[i])$, $T'[i] := (\widetilde{T}[i])^{\alpha'}$
Return $(\mathsf{pp} = (e, \mathbb{G}, \mathbb{G}_T, g, p, \widetilde{\mathsf{H}}, \overline{\mathsf{H}}, Y', A', \mathsf{pp}_{\mathsf{CH}}), \mathsf{ap} = T', msk = (\widetilde{T}, \mathsf{S}, s))$
```

```
$\underline{\mathsf{KG}(\mathsf{pp}, \mathsf{ap}):}$    $/\!/ n' = |T|$
$\alpha \leftarrow \mathbb{Z}_p^*$, $\beta \leftarrow \mathbb{Z}_p^*$, $X := g^{\beta}, Y := (Y')^{\beta}$
For each $i \in [n']$:    $T[i] := e(g, T'[i])^{\alpha}$
Return $(pk = (T, X, Y), sk = (\alpha, \beta))$

$\underline{\mathsf{Enc}(\mathsf{pp}, pk, x, m):}$
For each $j \in [k]$:
    $\gamma_j \leftarrow \mathbb{Z}_p^*$, $Q_j := e(A', \widetilde{\mathsf{H}}(x))^{\gamma_j}$
    $S_j := (T[\mathsf{H}_j(x)])^{\gamma_j} \cdot m$
If $e(X, Y') \neq e(g, Y)$: Return $\perp$
$r \leftarrow \mathbb{Z}_p^*$, $c := (g^r, e(\overline{\mathsf{H}}(x), Y)^r \cdot m)$
Return $\mathsf{ct} = ((Q_j, S_j)_{j \in [k]}, c)$
```

```
$\underline{\mathsf{TKGen}(\mathsf{pp}, msk, x):}$
$(\widetilde{T}, \mathsf{S}, s) \leftarrow msk$
Return $\mathsf{tk} := (\overline{\mathsf{H}}(x))^s$

$\underline{\mathsf{Dec}(\mathsf{pp}, sk, \mathsf{ct}, \mathsf{tk}):}$
$((Q_j, S_j)_{j \in [k]}, c, x) \leftarrow \mathsf{ct}$, $(\alpha, \beta) \leftarrow sk$
If $\mathsf{tk} \in \mathbb{G}$:
    $(U, V) \leftarrow c$
    Return $m := V / e(\mathsf{tk}^{\beta}, U)$
Else
    For $j \in [k]$:  $m_j := S_j \cdot Q_j^{-\alpha}$
    Return $\{m_1, \cdots, m_k\}$
```

$$\mathcal{R}_{\mathsf{ct}} = \{((\mathsf{pp}, pk, x, (Q_j, S_j)_{j \in [k]}, c = (U, V)), ((\gamma_j)_{j \in [k]}, r, m)) : \\ \wedge_{j \in [k]} (Q_j = e(A', \widetilde{\mathsf{H}}(x))^{\gamma_j} \wedge S_j = (T[\mathsf{H}_j(x)])^{\gamma_j} \cdot m) \wedge (U = g^r \wedge V = e(\overline{\mathsf{H}}(x), Y)^r \cdot m)\}$$

Fig. 5. A concrete USPCE scheme USPCE ($\mathcal{M} \subseteq \mathbb{G}_T$.)

description $(e, \mathbb{G}, \mathbb{G}_T, g, p)$. Then, we provide a concrete USPCE scheme USPCE as shown in Fig. 5.

We show how to prove the relation $\mathcal{R}_{\mathsf{ct}}$, and the concrete relation $\mathcal{R}_{\mathsf{ct}}$ is presented in Fig. 5. We can prove the well-formedness of S_j and V using the Sigma protocols, which can be found in the full version of this paper, while proving the well-formedness of Q_j and U using Schnorr's Sigma protocol [15]. Furthermore, applying the "AND-EQUAL$_l$" operations (the details of which can be found in the full version of this paper) over these sigma protocols, we can obtain a Sigma protocol for $\mathcal{R}_{\mathsf{ct}}$.

Remark 3. We can construct the algorithm $\mathsf{WellForm}_{\mathsf{tk}}$ mentioned in Remark 1 in this way: if $e(\mathsf{tk}, g) \neq e(\overline{\mathsf{H}}(x), Y')$, then return 0, otherwise return 1.

We analyze the correctness of USPCE as follows.

For any $\mathsf{S} \subset \mathcal{U}$, any $(\mathsf{pp}, \mathsf{ap}, msk) \leftarrow \mathsf{Setup}(\lambda, \mathsf{S})$, any $(pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp}, \mathsf{ap})$, and any $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{pp}, pk, x, m)$,

– when $x \in \mathsf{S}$, the properties of cuckoo hashing guarantee that x is inserted in one of locations (e.g., $\mathsf{H}_1(x), \ldots, \mathsf{H}_k(x)$) in T_{S}. Assuming x is located at

$\mathsf{H}_j(x)$ in the table, we obtain $\widetilde{\mathsf{H}}(x) = \widetilde{T}[\mathsf{H}_j(x)]$. Hence,

$$\begin{aligned}
&S_j \cdot Q_j^{-\alpha} \\
&\quad = (T[\mathsf{H}_j(x)])^{\gamma_j} \cdot m \cdot e(A', \widetilde{\mathsf{H}}(x))^{-\alpha\gamma_j} = (T[\mathsf{H}_j(x)])^{\gamma_j} \cdot m \cdot e(g^{\alpha'}, \widetilde{\mathsf{H}}(x))^{-\alpha\gamma_j} \\
&\quad = (T[\mathsf{H}_j(x)])^{\gamma_j} \cdot m \cdot e(g, (\widetilde{\mathsf{H}}(x))^{\alpha'})^{-\alpha\gamma_j} = (T[\mathsf{H}_j(x)])^{\gamma_j} \cdot m \cdot e(g, (\widetilde{T}[\mathsf{H}_j(x)])^{\alpha'})^{-\alpha\gamma_j} \\
&\quad = (T[\mathsf{H}_j(x)])^{\gamma_j} \cdot m \cdot e(g, T'[\mathsf{H}_j(x)])^{-\alpha\gamma_j} = (T[\mathsf{H}_j(x)])^{\gamma_j} \cdot m \cdot (T[\mathsf{H}_j(x)])^{-\gamma_j} = m
\end{aligned}$$

Hence, it is affirmed that $m \in S_{\mathrm{m}} = \{m_1, \cdots, m_k\}$, where $k = \lambda$ is a polynomial, indicating that the set S_{m} is polynomial.

– when $x \notin \mathsf{S}$, for $\mathsf{tk} \leftarrow \mathsf{TKGen}(\mathsf{pp}, msk, x)$, we obtain

$$\begin{aligned}
V/e(\mathsf{tk}^{\beta}, U) &= e(\overline{\mathsf{H}}(x), Y)^r \cdot m / e((\overline{\mathsf{H}}(x))^{s\beta}, g^r) = e(\overline{\mathsf{H}}(x), Y)^r \cdot m / e(\overline{\mathsf{H}}(x), (g^s)^{\beta})^r \\
&= e(\overline{\mathsf{H}}(x), Y)^r \cdot m / e(\overline{\mathsf{H}}(x), (Y')^{\beta})^r = e(\overline{\mathsf{H}}(x), Y)^r \cdot m / e(\overline{\mathsf{H}}(x), Y)^r = m
\end{aligned}$$

Security Analysis. Now, we show that the USPCE in Fig. 5 satisfies the aforementioned security requirements. Formally, we have the following theorem, the proof of which is given in the full version of this paper, due to the space limitations.

Theorem 1. USPCE *achieves confidentiality against authority, confidentiality against users, and confidentiality of sets.*

5 Dual HPS-KEM$^{\Sigma}$

In this section, we introduce a new primitive called *dual HPS-based KEM supporting Sigma protocols (dual* HPS-KEM$^{\Sigma}$*)*. We present a dual HPS-KEM$^{\Sigma}$ scheme based on the DDH assumption. Similar to [13], our scheme can also be extended to be based on the k-linear assumption [11,16].

Definition. A dual HPS-KEM$^{\Sigma}$ scheme $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ = ($\mathsf{KEMSetup}, \mathsf{KG}, \mathsf{CheckKey}, \mathsf{Encap}_{\mathrm{c}}, \mathsf{Encap}_{\mathrm{c}}^{*}, \mathsf{Encap}_{\mathrm{k}}, \mathsf{Decap}, \mathsf{dEncap}_{\mathrm{k}}, \mathsf{dDecap}, \mathsf{SamEncK}, \mathsf{dSamEncK}, \mathsf{CheckCwel}$) is a tuple of algorithms associated with a secret key space $\mathcal{SK}$, an encapsulated key space $\mathcal{K}$, and a tag space $\mathcal{T}$, where $\mathsf{Encap}_{\mathrm{c}}$, $\mathsf{Encap}_{\mathrm{k}}$ and $\mathsf{dEncap}_{\mathrm{k}}$ have the same randomness space $\mathcal{RS}$. We use $\mathcal{RS}^*$ to denote the randomness space of $\mathsf{Encap}_{\mathrm{c}}^{*}$.

- $\mathsf{pp} \leftarrow \mathsf{KEMSetup}(\lambda)$: On input a security parameter λ, it outputs a public parameter pp.
- $(pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp})$: On input the public parameter pp, it outputs a pair of public/secret keys (pk, sk).
- $b \leftarrow \mathsf{CheckKey}(\mathsf{pp}, sk, pk)$: On input the public parameter pp, a secret key sk and a public key pk, it outputs a bit b. Let $\mathcal{SK}_{\mathsf{pp},pk} := \{sk \in \mathcal{SK} \mid \mathsf{CheckKey}(\mathsf{pp}, sk, pk) = 1\}$.
- $c \leftarrow \mathsf{Encap}_{\mathrm{c}}(\mathsf{pp}; r)$: On input the public parameter pp with inner randomness $r \in \mathcal{RS}$, it outputs a well-formed ciphertext c. Let $\mathcal{C}_{\mathsf{pp}}^{\text{well-f}} := \{c = \mathsf{Encap}_{\mathrm{c}}(\mathsf{pp}; r) \mid r \in \mathcal{RS}\}$.

- $c \leftarrow \mathsf{Encap}^*_{\mathsf{c}}(\mathsf{pp}; r^*_{\mathsf{c}})$: On input the public parameter pp with inner randomness $r^*_{\mathsf{c}} \in \mathcal{RS}^*$, it outputs a ciphertext c. Let $\mathcal{C}^*_{\mathsf{pp}} := \{\mathsf{Encap}^*_{\mathsf{c}}(\mathsf{pp}; r^*_{\mathsf{c}}) \mid r^*_{\mathsf{c}} \in \mathcal{RS}^*\}$. We require $\mathcal{C}^{\text{well-f}}_{\mathsf{pp}} \subset \mathcal{C}^*_{\mathsf{pp}}$.
- $k \leftarrow \mathsf{Encap}_{\mathrm{k}}(\mathsf{pp}, pk; r)$: On input the public parameter pp and a public key pk with inner randomness $r \in \mathcal{RS}$, it outputs an encapsulated key $k \in \mathcal{K}$.
- $k' \leftarrow \mathsf{Decap}(\mathsf{pp}, sk, c)$: On input the public parameter pp, a secret key sk and a ciphertext c, and it outputs an encapsulated key $k' \in \mathcal{K}$.
- $k_{\mathsf{d}} \leftarrow \mathsf{dEncap}_{\mathrm{k}}(\mathsf{pp}, pk, t; r)$: On input the public parameter pp, a public key pk, and a tag $t \in \mathcal{T}$ with inner randomness $r \in \mathcal{RS}$, it outputs an encapsulated key $k \in \mathcal{K}$.
- $k'_{\mathsf{d}} \leftarrow \mathsf{dDecap}(\mathsf{pp}, sk, t, c)$: On input the public parameter pp, a secret key sk, a tag $t \in \mathcal{T}$ and a ciphertext c, it outputs an encapsulated key $k'_{\mathsf{d}} \in \mathcal{K}$.
- $k \leftarrow \mathsf{SamEncK}(\mathsf{pp}; r^*_{\mathsf{k}})$: On input the public parameter pp with inner randomness $r^*_{\mathsf{k}} \in \mathcal{RS}^*$, it outputs an encapsulated key $k \in \mathcal{K}$.
- $k_{\mathsf{d}} \leftarrow \mathsf{dSamEncK}(\mathsf{pp}, t; r^*_{\mathsf{k}})$: On input the public parameter pp and a tag $t \in \mathcal{T}$ with inner randomness $r^*_{\mathsf{k}} \in \mathcal{RS}^*$, it outputs an encapsulated key $k_{\mathsf{d}} \in \mathcal{K}$.
- $b \leftarrow \mathsf{CheckCwel}(\mathsf{pp}, c, r^*_{\mathsf{c}})$: On input the public parameter pp, a ciphertext c and a random number $r^*_{\mathsf{c}} \in \mathcal{RS}^*$, it outputs a bit b.

Correctness requirements are as follows.

(1) For any pp generated by $\mathsf{KEMSetup}(\lambda)$, and any (pk, sk) output by $\mathsf{KG}(\mathsf{pp})$, $\mathsf{CheckKey}(\mathsf{pp}, sk, pk) = 1$.
(2) For any pp generated by $\mathsf{KEMSetup}(\lambda)$, any (pk, sk) satisfying $\mathsf{CheckKey}(\mathsf{pp}, sk, pk) = 1$, any $t \in \mathcal{T}$, any randomness $r \in \mathcal{RS}$ and $c = \mathsf{Encap}_{\mathrm{c}}(\mathsf{pp}; r)$, it holds that $\mathsf{Encap}_{\mathrm{k}}(\mathsf{pp}, sk; r) = \mathsf{Decap}(\mathsf{pp}, sk, c)$, and $\mathsf{dEncap}_{\mathrm{k}}(\mathsf{pp}, pk, t; r) = \mathsf{dDecap}(\mathsf{pp}, sk, t, c)$.
(3) For any pp generated by $\mathsf{KEMSetup}(\lambda)$, and any c generated with $\mathsf{Encap}^*_{\mathsf{c}}(\mathsf{pp}; r^*_{\mathsf{c}})$, $\mathsf{CheckCwel}(\mathsf{pp}, c, r^*_{\mathsf{c}}) = 1$ if and only if $c \in \mathcal{C}^{\text{well-f}}_{\mathsf{pp}}$.

For any pp generated by $\mathsf{KEMSetup}(\lambda)$, we define some relations as follows:

$$\begin{aligned}
\mathcal{R}_{\mathrm{s}} &= \{(pk, sk) : \mathsf{CheckKey}(\mathsf{pp}, sk, pk) = 1\}, \mathcal{R}^*_{\mathrm{c}} = \{(c, r^*_{\mathsf{c}}) : c = \mathsf{Encap}^*_{\mathsf{c}}(\mathsf{pp}; r^*_{\mathsf{c}})\} \\
\mathcal{R}_{\mathrm{c,k}} &= \{((c, k, pk), r) : (c = \mathsf{Encap}_{\mathrm{c}}(\mathsf{pp}; r)) \wedge (k = \mathsf{Encap}_{\mathrm{k}}(\mathsf{pp}, pk; r))\} \\
\mathcal{R}^{\mathsf{d}}_{\mathrm{c,k}} &= \{((c, k_{\mathsf{d}}, pk), (t, r)) : (c = \mathsf{Encap}_{\mathrm{c}}(\mathsf{pp}; r)) \wedge (k_{\mathsf{d}} = \mathsf{dEncap}_{\mathrm{k}}(\mathsf{pp}, pk, t; r))\} \\
\mathcal{R}^*_{\mathrm{k}} &= \{(k, r^*_{\mathsf{k}}) : k = \mathsf{SamEncK}(\mathsf{pp}; r^*_{\mathsf{k}})\}, \mathcal{R}^{\mathsf{d}*}_{\mathrm{k}} = \{(k_{\mathsf{d}}, (t, r^*_{\mathsf{k}})) : k_{\mathsf{d}} = \mathsf{dSamEncK}(\mathsf{pp}, t; r^*_{\mathsf{k}})\}
\end{aligned} \tag{2}$$

We require that for each relation in Eq. (2), there is a Sigma protocol. Note that for $\mathcal{R}^{\mathsf{d}}_{\mathrm{c,k}}$ and $\mathcal{R}^{\mathsf{d}*}_{\mathrm{k}}$, we further require a *witness-only* Sigma protocol (please refer to the definition of witness-only in the full version of this paper)

We also require that $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ should satisfy the following properties.

Definition 15 (Universality). $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ *is* universal, *if for any computationally unbounded adversary* $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{univ}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) := \Pr[\mathbf{G}^{\mathrm{univ}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, *where* $\mathbf{G}^{\mathrm{univ}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$ *is defined in Fig. 6.*

```
$\mathbf{G}^{\text{univ}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$:
    $\mathsf{pp} \leftarrow \mathsf{KEMSetup}(\lambda)$, $(pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp})$
    $(c, k, r_{\mathsf{c}}^{*}) \leftarrow \mathcal{A}(\mathsf{pp}, pk)$
      s.t. $((c, r_{\mathsf{c}}^{*}) \in \mathcal{R}_{\mathsf{c}}^{*}) \wedge (\mathsf{CheckCwel}(\mathsf{pp}, c, r_{\mathsf{c}}^{*}) = 0)$
    If $k = \mathsf{Decap}(\mathsf{pp}, sk, c)$: Return 1
    Else Return 0

$\mathbf{G}^{\text{ex-univ}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$:
    $\mathsf{pp} \leftarrow \mathsf{KEMSetup}(\lambda)$, $(pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp})$
    $(c, k, r_{\mathsf{c}}^{*}, t) \leftarrow \mathcal{A}(\mathsf{pp}, pk)$
      s.t. $((c, r_{\mathsf{c}}^{*}) \in \mathcal{R}_{\mathsf{c}}^{*}) \wedge (\mathsf{CheckCwel}(\mathsf{pp}, c, r_{\mathsf{c}}^{*}) = 0)$
    If $k = \mathsf{dDecap}(\mathsf{pp}, sk, t, c)$: Return 1
    Else Return 0

$\mathbf{G}^{\text{C-unexpl}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$:
    $\mathsf{pp} \leftarrow \mathsf{KEMSetup}(\lambda)$,
    $(c, r_{\mathsf{c}}^{*}) \leftarrow \mathcal{A}(\mathsf{pp})$ s.t. $(c, r_{\mathsf{c}}^{*}) \in \mathcal{R}_{\mathsf{c}}^{*}$
    If $\mathsf{CheckCwel}(\mathsf{pp}, c, r_{\mathsf{c}}^{*}) = 1$: Return 1
    Else Return 0

$\mathbf{G}^{\text{K-unexpl}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$:
    $\mathsf{pp} \leftarrow \mathsf{KEMSetup}(\lambda)$, $(pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp})$
    $(c, r_{\mathsf{c}}^{*}, k, r_{\mathsf{k}}^{*}) \leftarrow \mathcal{A}(\mathsf{pp}, pk, sk)$
        s.t. $((c, r_{\mathsf{c}}^{*}) \in \mathcal{R}_{\mathsf{c}}^{*}) \wedge ((k, r_{\mathsf{k}}^{*}) \in \mathcal{R}_{\mathsf{k}}^{*})$
    If $\mathsf{Decap}(\mathsf{pp}, sk, c) = k$: Return 1
    Else Return 0

$\mathbf{G}^{\text{ex-K-unexpl}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$:
    $\mathsf{pp} \leftarrow \mathsf{KEMSetup}(\lambda)$, $(pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp})$
    $(c, r_{\mathsf{c}}^{*}, k_{\mathsf{d}}, t, r_{\mathsf{k}}^{*}) \leftarrow \mathcal{A}(\mathsf{pp}, pk, sk)$
        s.t. $((c, r_{\mathsf{c}}^{*}) \in \mathcal{R}_{\mathsf{c}}^{*}) \wedge ((k_{\mathsf{d}}, (t, r_{\mathsf{k}}^{*})) \in \mathcal{R}_{\mathsf{k}}^{\mathsf{d}*})$
    If $\mathsf{dDecap}(\mathsf{pp}, sk, t, c) = k_{\mathsf{d}}$: Return 1
    Else Return 0

$\mathbf{G}^{\text{sk-2pr}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$:
    $\mathsf{pp} \leftarrow \mathsf{KEMSetup}(\lambda)$, $(pk, sk) \leftarrow \mathsf{KG}(\mathsf{pp})$
    $sk' \leftarrow \mathcal{A}(\mathsf{pp}, pk, sk)$
    If $(sk' \neq sk) \wedge (\mathsf{CheckKey}(\mathsf{pp}, sk', pk) = 1)$:
        Return 1
    Return 0

$\mathbf{G}^{\text{ind}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$:
    $b \leftarrow \{0,1\}$, $\mathsf{pp} \leftarrow \mathsf{KEMSetup}(\lambda)$, $c_0 \leftarrow \mathsf{Encap}_{\mathsf{c}}(\mathsf{pp})$, $c_1 \leftarrow \mathsf{Encap}_{\mathsf{c}}^{*}(\mathsf{pp})$, $b' \leftarrow \mathcal{A}(\mathsf{pp}, c_b)$
    Return $(b' = b)$
```

Fig. 6. Games for $\mathsf{dHPS\text{-}KEM}^{\Sigma}$

Definition 16 (Extended universality). $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ *is* extended universal, *if for any computationally unbounded adversary* $\mathcal{A}$, $\mathbf{Adv}^{\text{ex-univ}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) := \Pr[\mathbf{G}^{\text{ex-univ}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, *where* $\mathbf{G}^{\text{ex-univ}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$ *is defined in Fig.* 6.

Definition 17 (Ciphertext unexplainability). $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ *is* ciphertext-unexplainable, *if for any PPT adversary* $\mathcal{A}$, $\mathbf{Adv}^{\text{C-unexpl}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) := \Pr[\mathbf{G}^{\text{C-unexpl}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, *where* $\mathbf{G}^{\text{C-unexpl}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$ *is defined in Fig.* 6.

Definition 18 (Key unexplainability). $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ *is* key-unexplainable, *if for any PPT adversary* $\mathcal{A}$, $\mathbf{Adv}^{\text{K-unexpl}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) := \Pr[\mathbf{G}^{\text{K-unexpl}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, *where* $\mathbf{G}^{\text{K-unexpl}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$ *is defined in Fig.* 6.

Definition 19 (Extended key unexplainability). $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ *is* extended key-unexplainable, *if for any PPT adversary* $\mathcal{A}$, $\mathbf{Adv}^{\text{ex-K-unexpl}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) := \Pr[\mathbf{G}^{\text{ex-K-unexpl}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, *where* $\mathbf{G}^{\text{ex-K-unexpl}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$ *is defined in Fig.* 6.

Definition 20 (Indistinguishability). $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ *is* indistinguishable, *if for any PPT adversary* $\mathcal{A}$, $\mathbf{Adv}^{\text{ind}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) := |\Pr[\mathbf{G}^{\text{ind}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$, *where* $\mathbf{G}^{\text{ind}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$ *is defined in Fig.* 6.

Definition 21 (SK-2PR). $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ *is* SK-second-preimage resistant, *if for any PPT adversary* $\mathcal{A}$, $\mathbf{Adv}^{\text{sk-2pr}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) := \Pr[\mathbf{G}^{\text{sk-2pr}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, *where* $\mathbf{G}^{\text{sk-2pr}}_{\mathsf{dHPS\text{-}KEM}^{\Sigma},\mathcal{A}}(\lambda)$ *is defined in Fig.* 6.

```
KEMSetup($\lambda$):
    pp $:= (\mathbb{G}, p, g_1, g_2) \leftarrow \mathcal{G}(\lambda)$
    Return pp

KG(pp):
    $(x_1, x_2) \leftarrow (\mathbb{Z}_p^*)^2$, $h := g_1^{x_1} g_2^{x_2}$
    Return $(pk = h,\ sk = (x_1, x_2))$

CheckKey(pp, $sk = (x_1, x_2), pk = h$):
    If $(g_1^{x_1} g_2^{x_2} = h)$: Return 1
    Else Return 0

Encap$_\mathsf{c}$(pp):
    $r \leftarrow \mathbb{Z}_p^*$
    Return $c := (u_1 = g_1^r,\ u_2 = g_2^r)$

Encap$_\mathsf{c}^*$(pp):
    $r_\mathsf{c}^* := (r, r') \leftarrow (\mathbb{Z}_p^*)^2$
    Return $c := (u_1 = g_1^r, u_2 = g_1^{r'})$

Encap$_\mathsf{k}$(pp, $pk = h$):
    $r \leftarrow \mathbb{Z}_p^*$
    Return $k := h^r$

SamEncK(pp):
    $r_\mathsf{k}^* := (r, r') \leftarrow (\mathbb{Z}_p^*)^2$
    Return $k := g_1^r g_2^{r'}$

dSamEncK(pp, $t \in \mathbb{G}$):
    $r_\mathsf{k}^* := (r, r') \leftarrow (\mathbb{Z}_p^*)^2$
    Return $k_\mathsf{d} := g_1^r g_2^{r'} \cdot t$

Decap(pp, $sk = (x_1, x_2), c = (u_1, u_2)$):
    Return $k' := u_1^{x_1} u_2^{x_2}$

dEncap$_\mathsf{k}$(pp, $pk = h, t \in \mathbb{G}$):
    $r \leftarrow \mathbb{Z}_p^*$
    Return $k_\mathsf{d} := h^r \cdot t$

dDecap(pp, $sk = (x_1, x_2), t, c = (u_1, u_2)$):
    Return $k_\mathsf{d}' := u_1^{x_1} u_2^{x_2} \cdot t$

CheckCwel(pp, $c = (u_1, u_2), r_\mathsf{c}^* = (r, r')$):
    If $(g_1^r = u_1) \wedge (g_2^r = u_2)$: Return 1
    Return 0
```

$\mathcal{R}_\mathrm{s} = \{(pk, (x_1, x_2)) : pk = g_1^{x_1} g_2^{x_2}\}$ $\quad$ $\mathcal{R}_\mathrm{c,k} = \{(((u_1, u_2), k, pk), r) : u_1 = g_1^r \wedge u_2 = g_2^r \wedge k = pk^r\}$

$\mathcal{R}_\mathrm{k}^* = \{(k, (r, r')) : k = g_1^r g_2^{r'}\}$ $\quad$ $\mathcal{R}_\mathrm{c}^* = \{((u_1, u_2), (r, r')) : u_1 = g_1^r \wedge u_2 = g_1^{r'}\}$

$\mathcal{R}_\mathrm{k}^{\mathsf{d}*} = \{(k_\mathsf{d}, (t, (r, r'))) : k_\mathsf{d} = g_1^r g_2^{r'} \cdot t\}$ $\quad$ $\mathcal{R}_\mathrm{c,k}^\mathsf{d} = \{(((u_1, u_2), k_\mathsf{d}, pk), (t, r)) : u_1 = g_1^r \wedge u_2 = g_2^r \wedge k_\mathsf{d} = pk^r \cdot t\}$

Fig. 7. Algorithm descriptions of a concrete $\mathsf{dHPS\text{-}KEM}^\Sigma$. There are Sigma protocols for relations $\mathcal{R}_\mathrm{s}$, $\mathcal{R}_\mathrm{c,k}$, $\mathcal{R}_\mathrm{c}^*$, $\mathcal{R}_\mathrm{c,k}^\mathsf{d}$, $\mathcal{R}_\mathrm{k}^*$ and $\mathcal{R}_\mathrm{k}^{\mathsf{d}*}$: Okamoto's Sigma protocol [14] for $\mathcal{R}_\mathrm{s}$ and $\mathcal{R}_\mathrm{k}^*$, the Chaum-Pedersen protocol [4] for $\mathcal{R}_\mathrm{c,k}$, Schnorr's Sigma protocol [15] for $\mathcal{R}_\mathrm{c}^*$, the Chaum-Pedersen protocol [4] and the Sigma protocol for $\mathcal{R}_\mathrm{c,k}^\mathsf{d}$ (which can be found in the full version of this paper and it requires "AND-EQUAL$_l$" operations), and Okamoto's Sigma protocol [14] and the Sigma protocol for $\mathcal{R}_\mathrm{k}^{\mathsf{d}*}$ (which also can be found in the full version of this paper, and the obtained Sigma protocol is witness-only).

Definition 22 (Smoothness). $\mathsf{dHPS\text{-}KEM}^\Sigma$ *is* smooth, *if for any fixed* pp *generated by* KEMSetup *and any fixed pk generated by* KG, $\Delta((c, k), (c, k')) \leq \mathsf{negl}(\lambda)$, *where* $c \leftarrow \mathsf{Encap}_\mathsf{c}^*(\mathsf{pp})$, $k \leftarrow \mathcal{K}$, $sk \leftarrow \mathcal{SK}_{\mathsf{pp},pk}$ *and* $k' = \mathsf{Decap}(\mathsf{pp}, sk, c)$.

Definition 23 (Extended smoothness). $\mathsf{dHPS\text{-}KEM}^\Sigma$ *is* extended smooth, *if for any fixed* pp *generated by* KEMSetup *and any fixed pk generated by* KG, $\Delta((c, k, t), (c, k', t)) \leq \mathsf{negl}(\lambda)$, *where* $c \leftarrow \mathsf{Encap}_\mathsf{c}^*(\mathsf{pp})$, $k \leftarrow \mathcal{K}$, $t \leftarrow \mathcal{T}$, $sk \leftarrow \mathcal{SK}_{\mathsf{pp},pk}$ *and* $k' = \mathsf{dDecap}(\mathsf{pp}, sk, t, c)$.

Definition 24 (Special extended smoothness). $\mathsf{dHPS\text{-}KEM}^\Sigma$ *is* special extended smooth, *if for any fixed* pp *generated by* KEMSetup *and any fixed* (pk, sk) *generated by* KG, $\Delta((c, k), (c, k')) \leq \mathsf{negl}(\lambda)$, *where* $c \leftarrow \mathsf{Encap}_\mathsf{c}^*(\mathsf{pp})$, $k \leftarrow \mathcal{K}$, $t \leftarrow \mathcal{T}$ *and* $k' = \mathsf{dDecap}(\mathsf{pp}, sk, t, c)$.

Definition 25 (Uniformity of sampled keys). $\mathsf{dHPS\text{-}KEM}^\Sigma$ *has* uniformity of sampled keys, *if for any* pp *generated by* KEMSetup *and any* $t \in \mathcal{T}$, *it holds that* $\Delta(k, k') = 0$ *and* $\Delta(k, k'') = 0$, *where* $k \leftarrow \mathcal{K}$, $k' \leftarrow \mathsf{SamEncK}(\mathsf{pp})$ *and* $k'' \leftarrow \mathsf{dSamEncK}(\mathsf{pp}, t)$.

Construction. Here, we present a concrete construction of dual HPS-KEM$^\Sigma$, which satisfies all the aforementioned security properties. Let $\mathcal{G}$ be a group generation algorithm, taking λ as input and outputting $(\mathbb{G}, p, g_1, g_2)$, where $\mathbb{G}$ is a prime-order group, p is the order of $\mathbb{G}$, and g_1, g_2 are two random generators of $\mathbb{G}$. Our construction $\mathsf{dHPS\text{-}KEM}^\Sigma$ is shown in Fig. 7.

It is clear that the construction in Fig. 7 satisfies *correctness*. The relations $\mathcal{R}_\mathrm{s}$, $\mathcal{R}_\mathrm{c,k}$, $\mathcal{R}_\mathrm{c}^*$, $\mathcal{R}_\mathrm{c,k}^\mathsf{d}$, $\mathcal{R}_\mathrm{k}^*$ and $\mathcal{R}_\mathrm{k}^{\mathsf{d}*}$ are presented in Fig. 7.

Remark 4. The algorithm $\mathsf{WellForm_u}$ mentioned in Remark 1 can be built in this way: given $sk = (x_1, x_2)$, if $pk \neq g_1^{x_1} g_2^{x_2}$, then return 0, otherwise return 1.

For security properties, we present the following theorem, the proof of which is given in the full version of this paper, due to space limitations.

Theorem 2. *The above scheme* $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ *achieves universality, extended universality, smoothness, extended smoothness, special extended smoothness, and uniformity of sampled keys. Furthermore,*

- $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ *is ciphertext-unexplainable, key-unexplainable and extended key-unexplainable under the DL assumption.*
- $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ *is indistinguishable under the DDH assumption.*
- $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ *is SK-second-preimage resistant under the DL assumption.*

6 General Construction of MAMF

In this section, we present a framework for constructing MAMF using USPCE and dual HPS-KEM$^{\Sigma}$.

Let $\mathsf{dHPS\text{-}KEM}^{\Sigma}$=($\mathsf{KEMSetup}, \mathsf{KG}, \mathsf{CheckKey}, \mathsf{Encap_c}, \mathsf{Encap^*_c}, \mathsf{Encap_k}, \mathsf{Decap}, \mathsf{dEncap_k}, \mathsf{dDecap}, \mathsf{SamEncK}, \mathsf{dSamEncK}, \mathsf{CheckCwel}$) be a dual HPS-KEM$^{\Sigma}$ scheme, where $\mathcal{RS}$ denotes the randomness space of $\mathsf{Encap_c}$, $\mathsf{Encap_k}$ and $\mathsf{dEncap_k}$, $\mathcal{RS}^*$ denotes the randomness space of $\mathsf{Encap^*_c}$, $\mathsf{SamEncK}$ and $\mathsf{dSamEncK}$, $\mathcal{T}$ denotes the tag space, and $\mathcal{K}$ denotes the encapsulated key space.

Let $\mathsf{USPCE} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{Enc}, \mathsf{TKGen}, \mathsf{Dec})$ be a USPEC scheme with a universe of elements $\mathcal{U}$, a token space $\mathcal{TK}$ and a message space $\mathcal{M}$.

Our generic MAMF scheme $\mathsf{MAMF} = (\mathsf{Setup}, \mathsf{KG_{Ag}}, \mathsf{KG_J}, \mathsf{KG_u}, \mathsf{Frank}, \mathsf{Verify}, \mathsf{TKGen}, \mathsf{Judge}, \mathsf{Forge}, \mathsf{RForge}, \mathsf{JForge})$ is presented in Fig. 8. It's worth noting that the message space of MAMF $\mathcal{M} = \mathsf{USPCE}.\mathcal{U}$ and $\mathsf{USPCE}.\mathcal{M} = \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{T}$.

We mainly introduce the algorithm Frank. It calls $\mathsf{Encap_c}$, $\mathsf{Encap_k}$, and $\mathsf{dEncap_k}$ of $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ to generate a well-formed ciphertext and encapsulated keys, respectively. Subsequently, it invokes Enc of USPCE to encrypt the tag used for the generation of the encapsulated key of the judge. Afterward, it utilizes a NIZK proof algorithm $\mathsf{NIZK}^{\mathcal{R}}.\mathsf{Prove}$ to create a NIZK proof. The relation $\mathcal{R}$ is defined as follows (which is also introduced in Fig. 1):

$$\begin{aligned}
\mathcal{R} = \{((\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}}, m), (sk_{\mathsf{s}}, t, r, r^*_{\mathsf{c}}, r^*_{\mathsf{k}}, r_{\mathsf{USPCE}})) :\\
((pk_{\mathsf{s}}, sk_{\mathsf{s}}) \in \mathcal{R}_{\mathsf{s}} \wedge (((c, k_{\mathsf{J}}, pk_{\mathsf{J}}), (t, r)) \in \mathcal{R}^{\mathsf{d}}_{\mathrm{c,k}} \wedge_{\mathrm{eq}} ((pk_{\mathsf{USPCE}}, m, c_{\mathsf{t}}), (t, r_{\mathsf{USPCE}})) \in \mathcal{R}_{\mathsf{ct}}))\\
\vee((c, r^*_{\mathsf{c}}) \in \mathcal{R}^*_{\mathsf{c}} \wedge (k_{\mathsf{r}}, r^*_{\mathsf{k}}) \in \mathcal{R}^*_{\mathsf{k}} \wedge ((pk_{\mathsf{USPCE}}, m, c_{\mathsf{t}}), (t, r_{\mathsf{USPCE}})) \in \mathcal{R}_{\mathsf{ct}})\\
\vee((c, r^*_{\mathsf{c}}) \in \mathcal{R}^*_{\mathsf{c}} \wedge ((k_{\mathsf{J}}, (t, r^*_{\mathsf{k}})) \in \mathcal{R}^{\mathsf{d}*}_{\mathsf{k}} \wedge_{\mathrm{eq}} ((pk_{\mathsf{USPCE}}, m, c_{\mathsf{t}}), (t, r_{\mathsf{USPCE}})) \in \mathcal{R}_{\mathsf{ct}}))\},
\end{aligned}$$

For the NIZK proof system $\mathsf{NIZK}^{\mathcal{R}} = (\mathsf{Prove}, \mathsf{Verify})$ utilized in Fig. 8, we construct it as follows. It's noteworthy that for every sub-relation (i.e., $\mathcal{R}_{\mathrm{s}}$, $\mathcal{R}^{\mathsf{d}}_{\mathrm{c,k}}$, $\mathcal{R}^*_{\mathrm{c}}$, $\mathcal{R}^*_{\mathrm{k}}$, $\mathcal{R}^{\mathsf{d}*}_{\mathrm{k}}$ and $\mathcal{R}_{\mathsf{ct}}$), the dual HPS-KEM$^{\Sigma}$ scheme and USPCE ensure the existence of a Sigma protocol. Utilizing the technique of trivially combining Sigma protocols for "AND/OR" operations [3, Sec. 19.7] and the "AND-EQUAL$_l$" operations, a new Sigma protocol for the relation $\mathcal{R}$ is obtained. Subsequently, employing the Fiat-Shamir transform, we derive a NIZK proof system $\mathsf{NIZK}^{\mathcal{R}} = (\mathsf{Prove}, \mathsf{Verify})$ for $\mathcal{R}$ in the random oracle model.

```
$\underline{\mathsf{Setup}(\lambda)}$: Return $\mathsf{pp} := \mathsf{pp}_{\mathsf{KEM}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{KEMSetup}(\lambda)$

$\underline{\mathsf{KG}_{\mathsf{Ag}}(\mathsf{pp}, \mathsf{S})}$: Return $(pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}}, sk_{\mathsf{Ag}}) := (\mathsf{pp}_{\mathsf{USPCE}}, \mathsf{ap}_{\mathsf{USPCE}}, msk_{\mathsf{USPCE}}) \leftarrow \mathsf{USPCE.Setup}(\lambda, \mathsf{S})$

$\underline{\mathsf{KG}_{\mathsf{J}}(\mathsf{pp}, pk_{\mathsf{Ag}}, \mathsf{ap}_{\mathsf{Ag}})}$:
$(pk'_{\mathsf{J}}, sk'_{\mathsf{J}}) \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{KG}(\mathsf{pp}_{\mathsf{KEM}})$, $(pk_{\mathsf{USPCE}}, sk_{\mathsf{USPCE}}) \leftarrow \mathsf{USPCE.KG}(\mathsf{pp}_{\mathsf{USPCE}}, \mathsf{ap}_{\mathsf{USPCE}})$
Return $(pk_{\mathsf{J}} = (pk_{\mathsf{USPCE}}, pk'_{\mathsf{J}}), sk_{\mathsf{J}} = (sk_{\mathsf{USPCE}}, sk'_{\mathsf{J}}))$

$\underline{\mathsf{KG}_{\mathsf{u}}(\mathsf{pp})}$: Return $(pk, sk) \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{KG}(\mathsf{pp}_{\mathsf{KEM}})$

$\underline{\mathsf{Frank}(\mathsf{pp}, sk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m)}$:
$(pk_{\mathsf{USPCE}}, pk'_{\mathsf{J}}) \leftarrow pk_{\mathsf{J}}$, $r \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{RS}$, $c \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{Encap}_{\mathsf{c}}(\mathsf{pp}_{\mathsf{KEM}}; r)$, $k_{\mathsf{r}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{Encap}_{\mathsf{k}}(\mathsf{pp}_{\mathsf{KEM}}, pk_{\mathsf{r}}; r)$
$t \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{T}$, $k_{\mathsf{J}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{dEncap}_{\mathsf{k}}(\mathsf{pp}_{\mathsf{KEM}}, pk'_{\mathsf{J}}, t; r)$, $r_{\mathsf{USPCE}} \leftarrow \mathsf{USPCE}.\mathcal{RS}$, $c_{\mathsf{t}} \leftarrow \mathsf{USPCE.Enc}(pk_{\mathsf{USPCE}}, m, t; r_{\mathsf{USPCE}})$
$x \leftarrow (sk_{\mathsf{s}}, t, r, \perp, \perp, r_{\mathsf{USPCE}})$, $y \leftarrow (\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}}, m)$, $\pi \leftarrow \mathsf{NIZK}^{\mathcal{R}}.\mathsf{Prove}(pk_{\mathsf{r}}, y, x)$
Return $\sigma \leftarrow (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}})$

$\underline{\mathsf{Verify}(\mathsf{pp}, pk_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m, \sigma)}$:
$(\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}}) \leftarrow \sigma$
$y \leftarrow (\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}}, m)$
If $\mathsf{NIZK}^{\mathcal{R}}.\mathsf{Verify}(pk_{\mathsf{r}}, \pi, y) = 0$:
    Return 0
If $\mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{Decap}(\mathsf{pp}_{\mathsf{KEM}}, sk_{\mathsf{r}}, c) = k_{\mathsf{r}}$:
    Return 1
Return 0

$\underline{\mathsf{TKGen}(\mathsf{pp}, sk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m)}$:
$\mathsf{tk} \leftarrow \mathsf{USPCE.TKGen}(\mathsf{pp}_{\mathsf{USPCE}}, msk_{\mathsf{USPCE}}, m)$
Return $\mathsf{tk}$

$\underline{\mathsf{Judge}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m, \sigma, \mathsf{tk})}$:
$(sk_{\mathsf{USPCE}}, sk'_{\mathsf{J}}) \leftarrow sk_{\mathsf{J}}$, $(\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}}) \leftarrow \sigma$, $y \leftarrow (\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}}, m)$
If $\mathsf{NIZK}^{\mathcal{R}}.\mathsf{Verify}(pk_{\mathsf{r}}, \pi, y) = 0$: Return 0
If $\mathsf{tk} \neq \perp$:
    $t' \leftarrow \mathsf{USPCE.Dec}(\mathsf{pp}_{\mathsf{USPCE}}, sk_{\mathsf{USPCE}}, c_{\mathsf{t}}, \mathsf{tk})$
    If $\mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{dDecap}(\mathsf{pp}_{\mathsf{KEM}}, sk'_{\mathsf{J}}, t', c) = k_{\mathsf{J}}$: Return 1
    Return 0
If $\mathsf{tk} = \perp$:
    $S_{\mathsf{t}} \leftarrow \mathsf{USPCE.Dec}(\mathsf{pp}_{\mathsf{USPCE}}, sk_{\mathsf{USPCE}}, c_{\mathsf{t}}, \perp)$
    For $t' \in S_{\mathsf{t}}$:
        If $\mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{dDecap}(\mathsf{pp}_{\mathsf{KEM}}, sk'_{\mathsf{J}}, t', c) = k_{\mathsf{J}}$: Return 1
    Return 0

$\underline{\mathsf{Forge}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m)}$:
$(pk_{\mathsf{USPCE}}, pk'_{\mathsf{J}}) \leftarrow pk_{\mathsf{J}}$, $r^*_{\mathsf{c}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{RS}^*$, $c \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{Encap}^*_{\mathsf{c}}(\mathsf{pp}_{\mathsf{KEM}}; r^*_{\mathsf{c}})$, $r^*_{\mathsf{k}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{RS}^*$
$k_{\mathsf{r}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{SamEncK}(pp; r^*_{\mathsf{k}})$, $t \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{T}$, $k_{\mathsf{J}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{K}$
$r_{\mathsf{USPCE}} \leftarrow \mathsf{USPCE}.\mathcal{RS}$, $c_{\mathsf{t}} \leftarrow \mathsf{USPCE.Enc}(pk_{\mathsf{USPCE}}, m, t; r_{\mathsf{USPCE}})$
$x \leftarrow (\perp, t, \perp, r^*_{\mathsf{c}}, r^*_{\mathsf{k}}, r_{\mathsf{USPCE}})$, $y \leftarrow (\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}}, m)$, $\pi \leftarrow \mathsf{NIZK}^{\mathcal{R}}.\mathsf{Prove}(pk_{\mathsf{r}}, y, x)$
Return $\sigma \leftarrow (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}})$

$\underline{\mathsf{RForge}(\mathsf{pp}, pk_{\mathsf{s}}, sk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m)}$:
$(pk_{\mathsf{USPCE}}, pk'_{\mathsf{J}}) \leftarrow pk_{\mathsf{J}}$, $r^*_{\mathsf{c}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{RS}^*$, $c \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{Encap}^*_{\mathsf{c}}(\mathsf{pp}_{\mathsf{KEM}}; r^*_{\mathsf{c}})$, $k_{\mathsf{r}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{Decap}(\mathsf{pp}_{\mathsf{KEM}}, sk_{\mathsf{r}}, c)$
$t \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{T}$, $r^*_{\mathsf{k}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{RS}^*$, $k_{\mathsf{J}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{dSamEncK}(pp, t; r^*_{\mathsf{k}})$
$r_{\mathsf{USPCE}} \leftarrow \mathsf{USPCE}.\mathcal{RS}$, $c_{\mathsf{t}} \leftarrow \mathsf{USPCE.Enc}(pk_{\mathsf{USPCE}}, m, t; r_{\mathsf{USPCE}})$
$x \leftarrow (\perp, t, \perp, r^*_{\mathsf{c}}, r^*_{\mathsf{k}}, r_{\mathsf{USPCE}})$, $y \leftarrow (\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}}, m)$, $\pi \leftarrow \mathsf{NIZK}^{\mathcal{R}}.\mathsf{Prove}(pk_{\mathsf{r}}, y, x)$
Return $\sigma \leftarrow (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}})$

$\underline{\mathsf{JForge}(\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, sk_{\mathsf{J}}, m)}$:
$(pk_{\mathsf{USPCE}}, pk'_{\mathsf{J}}) \leftarrow pk_{\mathsf{J}}$, $(sk_{\mathsf{USPCE}}, sk'_{\mathsf{J}}) \leftarrow sk_{\mathsf{J}}$, $r^*_{\mathsf{c}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{RS}^*$, $c \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{Encap}^*_{\mathsf{c}}(\mathsf{pp}_{\mathsf{KEM}}; r^*_{\mathsf{c}})$
$r^*_{\mathsf{k}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{RS}^*$, $k_{\mathsf{r}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{SamEncK}(pp; r^*_{\mathsf{k}})$, $t \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathcal{T}$, $k_{\mathsf{J}} \leftarrow \mathsf{dHPS\text{-}KEM}^{\Sigma}.\mathsf{dDecap}(\mathsf{pp}_{\mathsf{KEM}}, sk'_{\mathsf{J}}, t, c)$
$r_{\mathsf{USPCE}} \leftarrow \mathsf{USPCE}.\mathcal{RS}$, $c_{\mathsf{t}} \leftarrow \mathsf{USPCE.Enc}(pk_{\mathsf{USPCE}}, m, t; r_{\mathsf{USPCE}})$
$x \leftarrow (\perp, t, \perp, r^*_{\mathsf{c}}, r^*_{\mathsf{k}}, r_{\mathsf{USPCE}})$, $y \leftarrow (\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}}, m)$, $\pi \leftarrow \mathsf{NIZK}^{\mathcal{R}}.\mathsf{Prove}(pk_{\mathsf{r}}, y, x)$
Return $\sigma \leftarrow (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}})$
```

Fig. 8. Algorithms of MAMF

Correctness Analysis. For any signature $\sigma \leftarrow \mathsf{Frank}(\mathsf{pp}, sk_{\mathsf{s}}, pk_{\mathsf{r}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, m)$, we parse $\sigma = (\pi, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}})$, and let $y := (\mathsf{pp}, pk_{\mathsf{s}}, pk_{\mathsf{Ag}}, pk_{\mathsf{J}}, c, k_{\mathsf{r}}, k_{\mathsf{J}}, c_{\mathsf{t}}, m)$.

We first analyze the output of Verify as follows: (i) the correctness of $\mathsf{NIZK}^{\mathcal{R}}$ guarantees that $\mathsf{NIZK}^{\mathcal{R}}.\mathsf{Verify}(k_{\mathsf{r}}, \pi, y) = 1$; (ii) the correctness of $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ guarantees that $\mathsf{Decap}(\mathsf{pp}, sk_{\mathsf{r}}, c) = k_{\mathsf{r}}$. So, Verify will return 1.

Next, we analyze the output of Judge as follows: (i) the correctness of $\mathsf{NIZK}^{\mathcal{R}}$ guarantees that $\mathsf{NIZK}^{\mathcal{R}}.\mathsf{Verify}(k_{\mathsf{r}}, \pi, y) = 1$; (ii) the correctness of USPCE guarantees that $t = \mathsf{USPCE.Dec}(\mathsf{pp}_{\mathsf{USPCE}}, sk_{\mathsf{USPCE}}, c_{\mathsf{t}}, \mathsf{tk})$, where $c_{\mathsf{t}} \leftarrow \mathsf{USPCE.Enc}(pk_{\mathsf{USPCE}}, m, t; r_{\mathsf{USPCE}})$; (iii) the correctness of $\mathsf{dHPS\text{-}KEM}^{\Sigma}$ guarantees that $\mathsf{Decap}(pp, sk'_{\mathsf{J}}, t, c) = k_{\mathsf{J}}$. Therefore, Judge will also return 1.

In fact, the second point (ii) of the correctness of Judge should be divided into two cases as the definition of correctness in Sect. 3. It can be trivially guaranteed by the correctness of USPCE, so we omit the details here.

Security Analysis. We have the following theorem, the proof of which is placed in the full version of this paper, due to space limitations.

Theorem 3. *If the USPCE scheme* USPCE *satisfies the properties defined in Sect. 4, the dual HPS-KEM*$^\Sigma$ *scheme* $\mathsf{dHPS\text{-}KEM}^\Sigma$ *satisfies the properties defined in Sect. 5, and* $\mathsf{NIZK}^\mathcal{R} = (\mathsf{Prove}, \mathsf{Verify})$ *is a Fiat-Shamir NIZK proof system for* $\mathcal{R}$*, then our scheme* MAMF *achieves the properties defined in Sect. 3.2.*

Concrete Construction and Improvements. Plugging the concrete USPCE in Sect. 4 and the concrete dual HPS-KEM$^\Sigma$ in Sect. 5 into our framework, we obtain a concrete MAMF scheme. Notably, our concrete USPCE is based on the DBDH assumption, featuring a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and our concrete dual HPS-KEM$^\Sigma$ is based on the DDH assumption. To integrate them into our framework, we require that the concrete dual HPS-KEM$^\Sigma$ is built over $\mathbb{G}_T$.

In the full version of this paper, we present some improvements on the concrete MAMF. Because of the algebraic structure of our USPCE and dual HPS-KEM$^\Sigma$, there exist Sigma protocols proving the well-formedness of the USPCE ciphertext and of the encapsulated key for the judge simultaneously. Therefore, we let the franking algorithm directly call the encryption algorithm of USPCE to encrypt the encapsulated key for the judge, instead of the tag t, and then change the relation $\mathcal{R}$ accordingly. Furthermore, by proving the same statements in different sub-relations simultaneously, we propose an enhancement to the Sigma protocol for the relation $\mathcal{R}$. As a result, it reduces about 2/3 of the space overhead for the response of the Sigma protocol, which implies a smaller signature size. More exact comparison can be found in the full version of this paper, due to space limitations.

In our MAMF scheme, if the legislative agency has provided tokens for some messages, then the judge possesses the ability to identify all senders of these messages. However, it may not be suitable for all scenarios, as discussed in Introduction. We propose a solution in the full version of this paper, empowering the legislative agency to generate a one-time token for a specific MAMF signature and a specific message, such that the judge can carry out content moderation for that specific signature and specific message.

Acknowledgements. We want to express our sincere appreciation to the anonymous reviewers for their valuable comments and suggestions! Zhengan Huang and Gongxian Zeng are supported by The Major Key Project of PCL (PCL2023A09). Junzuo Lai is supported by National Natural Science Foundation of China under Grant Nos. 62472198 and U2001205, Guangdong Basic and Applied Basic Research Foundation (Grant No. 2023B1515040020). Jian Weng is supported by National Natural Science Foundation of China under Grant Nos. 62332007 and U22B2028, Major Program of Guangdong Basic and Applied Research Project under Grant No. 2019B030302008, Science and Technology Major Project of Tibetan Autonomous Region of China under Grant No. XZ202201ZD0006G, Guangdong Provincial Science and Technology Project under Grant No. 2021A0505030033, National Joint Engineering Research Center of Network Security Detection and Protection Technology, Guangdong Key Laboratory of Data Security and Privacy Preserving, Guangdong Hong Kong Joint Laboratory for

Data Security and Privacy Protection, and Engineering Research Center of Trustworthy AI, Ministry of Education.

References

1. Abelson, H., Anderson, R.J., Bellovin, S.M., Benaloh, J., Blaze, M., Diffie, W., Gilmore, J., Green, M., Landau, S., Neumann, P.G., Rivest, R.L., Schiller, J.I., Schneier, B., Specter, M.A., Weitzner, D.J.: Keys under doormats: mandating insecurity by requiring government access to all data and communications. J. Cybersecur. **1**(1), 69–79 (2015)
2. Bartusek, J., Garg, S., Jain, A., Policharla, G.V.: End-to-end secure messaging with traceability only for illegal content. In: EUROCRYPT 2023. pp. 35–66. Springer (2023)
3. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.5 (2020)
4. Chaum, D., Pedersen, T.P.: Wallet databases with observers. In: CRYPTO 1992. pp. 89–105. Springer (1992)
5. Facebook: Facebook messenger app (2016), https://www.messenger.com/
6. Facebook: Messenger secret conversations technical whitepaper (2016), https://fbnewsroomus.files.wordpress.com/2016/07/secret_conversations_whitepaper-1.pdf
7. FBI: Going dark https://www.fbi.gov/services/operational-technology/going-dark, accessed in January 2024
8. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run turing machines on encrypted data. In: CRYPTO 2013. pp. 536–553. Springer (2013)
9. Green, M., Kaptchuk, G., Van Laer, G.: Abuse resistant law enforcement access systems. In: EUROCRYPT 2021. pp. 553–583. Springer (2021)
10. Grubbs, P., Lu, J., Ristenpart, T.: Message franking via committing authenticated encryption. In: CRYPTO 2017. pp. 66–97. Springer (2017)
11. Hofheinz, D., Kiltz, E.: Secure hybrid encryption from weakened key encapsulation. In: CRYPTO 2007. pp. 553–571. Springer (2007)
12. Issa, R., Alhaddad, N., Varia, M.: Hecate: Abuse reporting in secure messengers with sealed sender. In: USENIX Security 2022. pp. 2335–2352 (2022)
13. Lai, J., Zeng, G., Huang, Z., Yiu, S.M., Mu, X., Weng, J.: Asymmetric group message franking: Definitions and constructions. In: EUROCRYPT 2023. pp. 67–97. Springer (2023)
14. Okamoto, T.: An efficient divisible electronic cash scheme. In: CRYPTO 1995. pp. 438–451. Springer (1995)
15. Schnorr, C.: Efficient identification and signatures for smart cards. In: CRYPTO 1989. pp. 239–252. Springer (1989)
16. Shacham, H.: A Cramer-Shoup Encryption Scheme from the Linear Assumption and from Progressively Weaker Linear Variants. Cryptology ePrint Archive, Report 2007/074 (2007)
17. Tarabay, J.: Australian government passes contentious encryption law. The New York Times (2018)
18. Tyagi, N., Grubbs, P., Len, J., Miers, I., Ristenpart, T.: Asymmetric message franking: Content moderation for metadata-private end-to-end encryption. In: CRYPTO 2019. pp. 222–250. Springer (2019)

Delegatable Anonymous Credentials from Mercurial Signatures with Stronger Privacy

Scott Griffy[1(✉)], Anna Lysyanskaya[1], Omid Mir[2], Octavio Perez Kempner[3], and Daniel Slamanig[4]

[1] Brown University, Providence RI, USA
{scott_griffy,anna_lysyanskaya}@brown.edu
[2] AIT Austrian Institute of Technology, Vienna, Austria
omid.mir@ait.ac.at
[3] NTT Social Informatics Laboratories, Tokyo, Japan
octavio.perezkempner@ntt.com
[4] Universität der Bundeswehr München, Munich, Germany
daniel.slamanig@unibw.de

Abstract. Delegatable anonymous credentials (DACs) enable a root issuer to delegate credential-issuing power, allowing a delegatee to take a delegator role. To preserve privacy, credential recipients and verifiers should not learn anything about intermediate issuers in the delegation chain. One particularly efficient approach to constructing DACs is due to Crites and Lysyanskaya (CT-RSA '19). In contrast to previous approaches, it is based on mercurial signatures (a type of equivalence-class signature), offering a conceptually simple design that does not require extensive use of zero-knowledge proofs. Unfortunately, current constructions of "CL-type" DACs only offer a weak form of privacy-preserving delegation: if an adversarial issuer (even an honest-but-curious one) is part of a user's delegation chain, they can detect when the user shows its credential. This is because the underlying mercurial signature schemes allows a signer to identify his public key in a delegation chain.

We propose CL-type DACs that overcome the above limitation based on a new mercurial signature scheme that provides adversarial public key class hiding which ensures that adversarial signers who participate in a user's delegation chain cannot exploit that fact to trace users. We achieve this introducing structured public parameters for each delegation level. Since the related setup produces critical trapdoors, we discuss techniques from updatable structured reference strings in zero-knowledge proof systems (Groth et al. CRYPTO'18) to guarantee the required privacy needs. In addition, we propose a simple way to realize revocation for CL-type DACs via the concept of revocation tokens. While we showcase this approach to revocation using our DAC scheme, it is generic and can be applied to any CL-type DAC system. Revocation is a vital feature that is largely unexplored and notoriously hard to achieve for DACs, thus providing it can help to make DAC schemes more attractive in practical applications.

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15485, pp. 296–325, 2025.
https://doi.org/10.1007/978-981-96-0888-1_10

Keywords: Anonymous credentials · delegatable credentials · mercurial signatures · revocation · Public key class-hiding

1 Introduction

Anonymous credentials (ACs) allow an authority (the issuer) to issue user credentials that can then be used for anonymous authentication. This primitive was envisioned by Chaum in [15] and later technically realized by Camenisch and Lysyanskaya in [12]. Importantly, in an AC scheme, a verifier and a user (also called a "credential holder") engage in a showing (also called a "proof" or "presentation") which proves to the verifier that the user has a valid credential. The scheme is *anonymous* if a user can show their credential multiple times in an unlinkable fashion. Intuitively, anonymity means that after verifying the credentials of two users, an adversary should not be able to tell if the credentials are both from a single user or from two different users.

Delegatable anonymous credentials (DACs) were introduced by Chase and Lysyanskaya [14]. As the name suggests, DAC schemes allow a root issuer to delegate their credential-issuing power to other "intermediate" issuers. This delegation allows any intermediate issuer to issue credentials on behalf of the root issuer (or possibly, re-delegate their issuing power), creating a *delegation chain* between the root issuer, the intermediate issuers, and the credential holder. Belenkiy, Camenisch, Chase, Kohlweiss, Lysyanskaya and Shacham showed how to realize DACs for arbitrarily long delegation chains [2].

Delegation alleviates the burden on the root issuer without revealing the root issuer's secrets to any other issuer, similar to a key hierarchy in a public key infrastructure (PKI) system. Unlinkability of DACs ensures the anonymity of credential holders, as well as the anonymity of any issuers who participated in that credential's delegation chain. The anonymity of intermediate issuers implies that given the showing of two credentials, an adversarial verifier cannot determine if they were issued by the same intermediate issuer or different intermediate issuers. Hiding the intermediate issuer is important for a DAC scheme as revealing the identities of these intermediate issuers might reveal information about the end user. The root issuer is always identified in a showing as the verifier must trust some key for unforgeability.

An important property when practically using ACs and DACs is non-transferability. This property ensures that users cannot easily share their credentials with other users. One way of providing this is to ensure that a user cannot share one of her credentials without sharing all of her credentials (and corresponding secrets). This is known as the "all-or-nothing" approach [12] and should disincentivize sharing of credentials by users' fear of losing control over their credentials. Another feature that is particularly important for the practical application of (delegatable) anonymous credentials is revocation. Unfortunately, this property is often neglected. It is quite clear that when preserving user's privacy, standard approaches to recovation known from classical PKI schemes do not work. While there are various different approaches to revocation of anonymous credentials

[7,9–13,19], in the delegatable setting this seems much harder to achieve and the topic is largely unexplored [1].

1.1 Previous Work on DAC and Motivation for Our Work

As in the recent work of Mir, Slamanig, Bauer, and Mayrhofer [28], we are particularly interested in developing practical DAC schemes. For a broader understanding, readers are directed to their comprehensive overview. The first practical DAC was proposed by Camenisch, Drijvers, and Dubovitskaya [8], but unfortunately they do not support an anonymous delegation phase. This, however is a crucial privacy requirement. The DACs by Blömer and Bobolz [4] as well as [28] represent two relevant and efficient DAC candidates as they have anonymous delegations and additionally, compact credentials. Unfortunately, [28] does not provide the important property of non-transferability, and for both [4,28] the delegated credential is distributed independently of any of the previous delegators. Consequently, it seems very hard to efficiently achieve revocation of delegators for those schemes.

Crites and Lysyanskaya [17] came up with a simple architecture (which we will call "CL-type DAC") for delegatable anonymous credentials that uses *mercurial signatures* (MS). These CL-type DACs bring the use of equivalence class signatures, extensively used in anonymous credentials [16,19,20,25,26,28], to the DAC setting (with numerous follow up works [18,27,29,31] on various aspects). In CL-type DACs, the "links" in a delegation chain are signatures; this chain includes the root's signature on the first intermediate issuer's public key; then for $i \geq 1$, the i^{th} intermediary's signatures on the $i+1^{st}$ intermediary's public key, and finally the last intermediate issuer's signature on the credential holder's public key. In order to ensure unlinkability, mercurial signatures allow randomization of both the signer's public key to an equivalent but unlinkable public key, and the randomization of the message to an equivalent but unlinkable message. As an example delegation chain in a CL-type DAC, would first require a root (or certification) authority (CA) which holds a signing key of an MS scheme and to issue a credential to a user, Alice. To do this, the root authority produces a signature $\sigma_{CA,A}$ on an MS public key pk_A of Alice. By demonstrating knowledge of the corresponding secret key to pk_A along with the root's signature on pk_A, Alice can authenticate herself. Now if Alice wants to delegate a credential to Bob, she uses her corresponding secret key to produce a signature $\sigma_{A,B}$ on Bob's MS public key pk_B, where the signature acts as a credential for Bob. Now Bob can authenticate himself by demonstrating knowledge of the corresponding secret key (to pk_B) and showing the signatures from both the root ($\sigma_{CA,A}$) and from Alice ($\mathsf{pk}_A, \sigma_{A,B}$). This principle can be applied to an arbitrarily long delegation chain. Now assume that Bob wants to show in a privacy-preserving way that he has been delegated a credential by CA. He can do this by demonstrating $(\mathsf{pk}_{CA}, {\mathsf{pk}_A}', {\mathsf{pk}_B}'), (\sigma'_{CA,A}, \sigma'_{A,B})$ where ${\mathsf{pk}_A}'$ and ${\mathsf{pk}_B}'$ are new representatives of the respective key classes (and by the properties of MS, they are unlinkable to the previous ones) and the signatures $(\sigma'_{CA,A}, \sigma'_{A,B})$ are adapted to the new messages and public keys respectively (which are similarly unlinkable). In the concrete CL construction [17], the MS scheme works in a bilinear group setting where w.l.o.g. the public key of the CA lives in the second source group, $\mathbb{G}_2$, and the public key of Alice in $\mathbb{G}_1$. Consequently, since the public keys of the MS scheme on the next level need to live in the message space of the MS scheme of the previous level, one always needs to switch the groups for the MS scheme on every level, which is an important detail to keep in mind.

One important limitation of existing DAC approaches [17,18,28,31], besides not yet supporting revocation, is that they only satisfy a weak notion of privacy. In particular it is not possible to guarantee anonymity even in the case of an honest-but-curious delegator in the credential chain (or when the root authority and a single delegator on the delegation chain collaborate, in the case of [28]). In prior constructions of CL-type credentials [17,18,31] this is because public keys in these constructions are traceable (using the secret key) regardless of how they have been randomized. Thus, a malicious delegator can identify itself on a chain and break anonymity.

When it comes to revocation in DAC, the only work so far is by Acar and Nguyen [1], which is based on the generic DAC template in [2] from randomizable NIZK proofs and in addition uses homomorphic NIZK proofs. While this can be instantiated from the Groth-Sahai [24] proof system, this is not very attractive from a practical perspective due to significant costs. So having practical DACs with revocation is an open problem.

Consequently, there exists a gap in that we do not have practical DAC schemes with strong privacy guarantees that support practical revocation of delegators. Our aim is to close this gap.

Our work can be seen as the continuation of the research initiated by Crites and Lysyanskaya in [17], closing these existing gaps for practical DAC schemes. To do this, we create a mercurial signature scheme with a stronger privacy property called adversarial public key class-hiding. An overview of this approach is outlined in the technical overview in Sect. 1.3. Ensuring that maliciously created public keys in mercurial signatures are not traceable by their owners after being randomized has been an open problem since their introduction in [17]. A very recent work introduced the notion of interactive threshold mercurial signatures [29] to overcome said limitation, but it requires an interactive signing protocol that computes a signature from shares of a secret key that are distributed among parties. While such an approach is satisfactory for anonymous credentials and can also be used to distribute trust of the root authority in DAC schemes, it's unclear how it can be used to efficiently manage delegations. Instead, we introduce structured public parameters which we carefully glue over the delegation levels to enable strong privacy features (and without requiring any interaction). Since the setup of these parameters also produces trapdoors that endanger privacy, we show how to overcome this problem by using techniques well-known from updatable structured reference string in zero-knowledge proof systems [23]. For revocation in DAC, we introduce a new and practical approach that is applicable to any CL-type DAC scheme.

1.2 Our Contributions

Subsequently, we summarize our contributions:

New Mercurial Signatures. First, as our core building block, we define a new flavor of a mercurial signature scheme which satisfies a stronger class hiding property, namely *adversarial public-key class hiding* (APKCH). Unlike in the mercurial signature definition of Crites and Lysyanskaya [17], here the adversary comes up with a public key and signs a message of its choice; the challenge for an adversary is to distinguish between a randomized version of his own public key and signature and a fresh, unrelated public key and a signature on the same message under that fresh public key. We give a construction of an APKCH signature scheme in the generic group model. Adversarial public-key class hiding is sufficient to construct a DAC scheme with strong privacy.

New Technique for Revocation in DAC. We introduce a new revocation approach for DACs. The basic idea is that a revocation authority maintains a public deny list, which verifiers can use to ensure that a credential shown to them does not contain any revoked delegators. Thereby any user who wants to receive a credential or obtain delegation rights (while still supporting later revocation) must first register their key with the revocation authority. This registration is anonymous and neither a verification of the user's identity is needed nor a proof of knowledge of their key needs to be performed. This gives a simple privacy-preserving way for revocation in DAC, can be used for any CL-type DAC and does so without resorting to heavy tools such as malleable NIZKs as done in [1].

Model and Instantiation of DAC with Strong Privacy and Revocation. We introduce a conceptually simple model for revocable DAC schemes with strong privacy using game-based security definitions. We believe that this notion is easier to use than the simulation-based security notions provided in [1]. Then, using our mercurial signature scheme with $\ell = 2$, we construct a conceptually simple DAC scheme with delegator revocation and without requiring extensive use of zero-knowledge proofs. We stress that this gives a CL-type DAC scheme, where privacy holds even when the adversary is allowed to corrupt the root and all delegators simultaneously.

1.3 Technical Overview

Public keys in the mercurial signature from [17] are ℓ-length vectors of group elements and are constructed as $\mathsf{pk} = \{\hat{X}_i\}_{i\in[\ell]} = \{\hat{P}^{x_i}\}_{i\in[\ell]}$ where the secret key is $\mathsf{sk} = \{x_i\}_{i\in[\ell]} \in (\mathbb{Z}_p^*)^\ell$ and $\hat{P}$ is the generator the second source group of a bilinear pairing. Anyone can randomize a public key pk to a new representative of the equivalence class to get $\mathsf{pk}' = \mathsf{pk}^\rho = \{\hat{X}_i^\rho\}_{i\in[\ell]}$ for $\rho \in \mathbb{Z}_p$. Unfortunately, such public keys are immediately recognizable to an adversary who holds the corresponding secret key. For an adversary to recognize a public key, it suffices to exponentiate each element in the public key by the inverse of the corresponding element in the secret key and check that the result has the form: $\{\hat{P}^\rho\}_{i\in[\ell]}$ (a vector of equal elements).

One approach to overcome the above limitation is to ensure that each element in a public key is computed over a distinct generator of the group where the discrete logarithms between these generators are random and not known. For example, if we add a trusted setup to the scheme from [17]: $\mathsf{Setup}(1^\lambda) \rightarrow \mathsf{pp} = \{\hat{P}_1, \hat{P}_2, ..., \hat{P}_\ell\}$ where $\hat{P}_i = \hat{P}^{\hat{b}_i}$ for ℓ randomly sampled $\hat{b}_i$ scalars in $\mathbb{Z}_p$ and public keys are computed as $\mathsf{pk} = \{\hat{X}_i = \hat{P}_i^{x_i}\}_{i\in[\ell]}$ then it can be shown that under the DDH assumption that an adversary cannot distinguish a randomization of their public key from a freshly sampled key.

This appears promising, especially since these $\hat{P}_i$ values are all distinct and can be efficiently sampled in the ROM. But, if an honest user receives a public key, it is not immediately clear how to ensure that it wasn't created maliciously so that they are recognizable (e.g., ensure the adversary did not compute the public key independent of the public parameters as $\mathsf{pk} = \{\hat{P}^{x_i}\}_{i\in[\ell]}$ which would be recognizable). To ensure that maliciously created public keys are computed over these bases without the need for zero knowledge proofs, we add what we call "verification bases" to the public parameters. The verification bases are structured so that they can be paired with the key to prove that it was computed using the trusted public parameters. To accomplish this, we need to expand the size of the public key vectors to double their length (2ℓ). This extra half of the public key will be the result of exponentiating different bases in the public parameters with the same secret key as in the first half. Specifically, our public parameters (w.r.t. public keys) consist of $\hat{\mathbf{B}} = \{\hat{B}_i\}_{i\in[2\ell]} = \{\hat{P}^{\hat{b}_1}, \ldots, \hat{P}^{\hat{b}_\ell}, \hat{P}^{\hat{b}_1\hat{d}_1}, \ldots, \hat{P}^{\hat{b}_\ell\hat{d}_\ell}\}$ such that $\mathsf{dlog}_{\hat{B}_i}(\hat{B}_{\ell+i}) = \hat{d}_i$. We then include the verification bases in the public parameters which are computed as: $\mathbf{V} = \{V_i\}_{i\in[\ell]} = \{P^{v_1\hat{d}_1}, \ldots, P^{v_\ell\hat{d}_\ell}, P^{v_1}, \ldots, P^{v_\ell}\}$ (for randomly sampled scalars, $\{v_i\}_{i\in[\ell]}$) such that $\forall i \in [\ell], e(V_i, \hat{B}_i) = e(V_{\ell+i}, \hat{B}_{\ell+i})$. Then, key pairs are computed as $\mathsf{sk} = \{x_i\}_{i\in[\ell]} \leftarrow\$\, \mathbb{Z}_p$, $\mathsf{pk} = \{\hat{X}_i\}_{i\in[\ell]} = \{\hat{B}_i^{x_i}\}_{i\in[\ell]} \| \{\hat{B}_{\ell+i}^{x_i}\}_{i\in[\ell]}$. We can see now that $\forall i \in [\ell], e(V_i, \hat{X}_i) = e(V_{\ell+i}, \hat{X}_{\ell+i})$. Thus, a verifier can take this length 2ℓ public key and use the verification bases ($\mathbf{V}$) to verify it by computing these pairings. If these pairing equations hold, then, because the elements in the upper half of $\hat{\mathbf{B}}$ are the only elements in the second source group that are exponentiated with $\hat{d}_i$, the adversary must have computed the upper half of the public key with these bases. Through a similar argument, the lower half of the public key must be computed over the lower half of the public key bases in the public parameters. Thus, if these pairing equations hold for a public key, then randomizations of that public key are unrecognizable to the adversary. Because we need correlated randomness in the trapdoors (e.g. both V_i and $\hat{B}_{\ell+i}$ are computed using $\hat{d}_i$) we can no longer use the ROM to generate these parameters and instead must use the common reference string (CRS) model.

We quickly run into a second problem as with these modified public keys as correctness falls apart when we attempt to sign messages. In [17], signatures are computed as $\sigma = (Z, Y, \hat{Y})$ with $Z = (\prod_{i\in[\ell]} M_i^{x_i})^y$, $Y = P^{1/y}$, and $\hat{Y} = \hat{P}^{1/y}$. Verification is done via a pairing product equation: $e(Z, \hat{Y}) = \prod_{i\in[\ell]} e(M_i, \hat{X}_i)$, which will not verify with the new structure of public keys that we've introduced in this section. Therefore, we expand the message space to vectors of 2ℓ elements instead of ℓ elements and include $\forall i \in [\ell], P_i = P^{\hat{b}_i}$ in the public parameters. Messages then have the form: $M = \{P^{m_i}\}_{i\in[\ell]} || \{P_i^{m_i}\}_{i\in[\ell]}$ for some vector $\{m_i\}_{i\in[\ell]} \in \mathbb{Z}_p^{\ell}$. In this modified scheme, the structure of Y and $\hat{Y}$ remains unchanged, but we then use the upper half of the message vector in our Sign function and the lower half of the message vector in our Verify function. This modification leads to a correct verification, now given by: $e(Z, \hat{Y}) = e(P, \hat{P})^{\sum_{i\in[\ell]} m_i x_i \hat{b}_i} = \prod_{i\in[\ell]} e(M_i, \hat{X}_i)$. We also have to add more structure to achieve a signature that yields DAC as the lower half of the message (which will be another public key in a DAC credential chain) is recognizable. We add this extra structure in Sect. 3.

Constructing a Strongly Private DAC. As previously mentioned, [17] works by alternating the use of two signature schemes so that even delegation levels are signed with one of the schemes and the odd ones with the other. This way, the message space in one of the schemes is the public key space of the other.

Our approach is to build a structured random string so that each scheme can sign public keys for the next level of the credential chain using unique blinding factors taken from the CRS for each level. This poses a challenge as we need to correlate the structure of both schemes for messages and public keys. To this end, the keys used in our scheme are twice the size of the keys in [17]. Fortunately, for CL-type DACs $\ell = 2$ and typical applications that use delegation do not require long delegation chains (e.g., driving licenses or official identity documents), making this approach entirely practical. To illustrate it, we consider a DAC scheme for $L = 3$. We can generate the public parameters, $\mathsf{pp} = \{P^{\hat{b}_0}, P^{\hat{b}_1}\}$, $\mathsf{pp}' = \{\hat{P}^{\hat{b}'_0}, \hat{P}^{\hat{b}'_1}, \hat{P}^{\hat{b}_0\hat{b}'_0}, \hat{P}^{\hat{b}_1\hat{b}'_1}\}$, $\mathsf{pp}^* = \{P^{\hat{b}^*_0}, P^{\hat{b}^*_1}, P^{\hat{b}'_0\hat{b}^*_0}, P^{\hat{b}'_1\hat{b}^*_1}\}$. We can see that the bases from pp and pp' have a structure that satisfies: $e(P^{\hat{b}_i}, \hat{P}^{\hat{b}'_i}) = e(P, \hat{P}^{\hat{b}_i\hat{b}'_i})$ and similar for pp' and pp^*. Hence, such public parameters can be used to build public keys for the credential chain as: $\mathsf{pk} = \{P^{\hat{b}_0 x_0}, P^{\hat{b}_1 x_1}\}$, $\mathsf{pk}' = \{\hat{P}^{\hat{b}'_0 x'_0}, \hat{P}^{\hat{b}'_1 x'_1}, \hat{P}^{\hat{b}_0\hat{b}'_0 x'_0}, \hat{P}^{\hat{b}_1\hat{b}'_1 x'_1}\}$, $\mathsf{pk}^* = \{P^{\hat{b}^*_0 x^*_0}, P^{\hat{b}^*_1 x^*_1}, P^{\hat{b}'_0\hat{b}^*_0 x^*_0}, P^{\hat{b}'_1\hat{b}^*_1 x^*_1}\}$. It follows from inspection that if we use $\mathsf{sk} = \{x_0, x_1\}$ to sign the third and fourth elements of pk', the signature will verify using the first and second elements from pk'. Similarly, if we use $\mathsf{sk}' = \{x'_0, x'_1\}$ to sign the third and fourth elements in pk^*, the signature will verify under the first half of pk' with the first half of pk^* as the message. Because these trapdoors are shared across schemes, we need the security properties of our signature scheme to hold when multiple correlated copies of the scheme are generated. We describe this requirement of our signature scheme further in Sect. 3.3 where we present the above example with more detail in Fig. 5.

Removing Trust in the Parameter Generation. As it is apparent from our above discussion, the generation of parameters in setup involves a number of exponents that must not be available to any party. Putting trust in the party running this setup to discard these values is typically not desirable, especially in a DAC setting where there are numerous parties involved. One way to deal with this issue is to run specific multi-party computation protocols to generate the parameters [3,6], e.g., running distributed key generation protocols. In order to avoid interaction among many parties, one particularly appealing approach was proposed by Groth et al. [23] in the context of zk-SNARKs, i.e., so called updatable reference strings. This means that some (potentially malicious) party can generate a reference string and any (potentially malicious) party in the system can update the reference string. Thereby every party outputs a proof that the computation was performed honestly and when the chain of proofs from the generation until the last update of the reference string verifies and at least one of the involved parties is honest, it is ensured that no one knows the trapdoors. Since this process can be done strictly sequentially this seems much more interesting for practical application, as for instance demonstrated by the powers of tau ceremony recently run by Ethereum[1], with around 95k contributors (cf. [30]). We note that in our case this can be done very efficiently using Fiat-Shamir transformed Schnorr proofs for discrete logarithm relations. In the full version of this paper [22] we present the concrete relations for the updates. In brief, the costs are 5ℓ Pedersen commitments for the trapdoors, 8ℓ group elements for the Schnorr proofs for the base group elements in pp and 10ℓ $\mathbb{Z}_p$ elements for the Schnorr proofs which include the trapdoors. Concretely, for $\ell = 2$ this amounts to 26 group elements and 20 $\mathbb{Z}_p$ elements per level (where for usual applications $L \leq 5$) being several orders of magnitude smaller than the one run by Ethereum. We also note that the computation and verification of these Schnorr proofs is highly efficient.

Revoking Intermediate Issuers and Users in a DAC Scheme. Conceptually (but not technically) our approach to revocation shares similarities with verifier local revocation kown from group signatures [5]. In particular, to revoke these credentials, we generate revocation tokens that verify with a given public key and can later be provided to a trusted revocation authority (TRA). The TRA adds these tokens to a deny list. To achieve this, the TRA first creates the ephemeral secret and public keys. The TRA then registers the user by signing the user's public keys with the corresponding ephemeral secret key. This forms a signature chain similar to the mercurial scheme described in [17].

When a user needs to prove their credentials, they present a revocation token for each public key in their chain. Since this revocation method mirrors the credential chain approach in [17], i.e., mercurial signatures on public keys, the tokens can be randomized to maintain user anonymity during the presentation.

To revoke a user or issuer in a credential chain (perhaps if the credential chain is used to perform some illegal action) these revocation tokens can be supplied from the verifier to the TRA who can then look through all the secret ephemeral keys they generated to recognize the credential chain and add the respective ones to a deny list.

Later, any verifier can verify the TRA's signature in the revocation token and iterate through the deny list, using each secret key to attempt to match each public key in the chain. More specifically, the verifier exponentiates the ephemeral public key in the revocation token with the inverses of the secret keys

[1] https://github.com/ethereum/kzg-ceremony-specs.

in the deny list (as described earlier in Sect. 1.3). If a match is found, the verifier can confirm that the credential has been revoked (cf. Section 4 for details).

2 Background

Notation. We use $[\ell]$ to denote the set, $\{1, 2, \ldots, \ell\}$. We use the notation $x \in \mathsf{Func}$ to mean that x is a possible output of the function, Func. When drawing multiple values from a set, we may omit notation for products of sets, e.g. $(x, y) \in \mathbb{Z}_p$ is the same as $(x, y) \in (\mathbb{Z}_p)^2$ where only the latter is formally correct. For a map from the set Z to the set S, $m : Z \rightarrow S$, we will denote $m[i] \in S$ as the output of the map in S with input $i \in Z$. We use bold font to denote a vector (e.g. $\mathbf{V}$). For brevity, we will sometimes denote the elements in a vector as $\mathbf{V} = \{V_i\}_{i\in[\ell]} = \{V_1, \ldots, V_\ell\}$. For a vector, $\mathbf{V} = \{V_1, \ldots, V_\ell\}$, of group elements, we denote the exponentiation of each element by a scalar ($\rho \in \mathbb{Z}_p$) with the notation: $\mathbf{V}^\rho = \{V_1^\rho, \ldots, V_\ell^\rho\}$. We use "wildcards" $(*)$ in equations. For example, the equation $(a, b) = (a', *)$, holds if $a = a'$ no matter what the value of b is. By $(m, *) \in S$ we mean there is a tuple in the set S such that the first element of the tuple is m and the second element is another value which could be anything. $\{(m, *) \in S : A(m)\}$ is the set of all tuples in S with m as their first element meeting condition A. For two distributions, A and B, we use the notation, $A \sim B$, to denote that they are computationally indistinguishable.

2.1 Bilinear Pairings

A bilinear pairing is a set of cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p, along with a pairing function, e (where $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$) which preserves structure. We call $\mathbb{G}_T$ the "target group" and call $\mathbb{G}_1$ and $\mathbb{G}_2$ the first and second "source groups" respectively. In this work, we use Type III pairings, which means that there is no efficient, non-trivial homomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$. The pairing function is efficiently computable and has a bilinearity property such that if $\langle P \rangle = \mathbb{G}_1$ and $\langle \hat{P} \rangle = \mathbb{G}_2$, then for $a, b \in \mathbb{Z}_p^*$, $e(P^a, \hat{P}^b) = e(P, \hat{P})^{ab}$. In our pairing groups, the Diffie-Hellman assumptions hold in the related groups, such that for $a, b, c \leftarrow\$\, \mathbb{Z}_p$, $(P^a, P^b, P^{ab}) \sim (P^a, P^b, P^c)$. Also, given (P^a, P^b) it is difficult to compute P^{ab}. We'll use hats to denote elements in the second source group, e.g. $\hat{X} \in \mathbb{G}_2, X \in \mathbb{G}_1$. We also use the generic group model in the bilinear pairing setting [33].

2.2 Mercurial Signatures

The original scheme from [17] comprises the following algorithms: Setup, KeyGen, Sign, Verify, $\mathsf{ConvertPK}$, $\mathsf{ConvertSK}$, $\mathsf{ConvertSig}$, and $\mathsf{ChangeRep}$. The scheme is parametrized by a length, ℓ, which determines the upper bound on the size of messages that can be signed. A mercurial signature scheme is parameterized by equivalence relations for the message, public key, and secret key spaces: $\mathcal{R}_M$, $\mathcal{R}_{\mathsf{pk}}$, $\mathcal{R}_{\mathsf{sk}}$. These relations form *equivalence classes* for messages and keys and define exactly how messages and signatures can be randomized such that their corresponding signatures can correctly be updated to verify with the updated keys and messages. Allowing the keys and messages to be randomized is what gives this signature scheme its privacy-preserving properties. In this work, we introduce auxiliary algorithms to verify the correctness of messages and public keys with respect to the scheme parameters. These are VerifyMsg and VerifyKey, respectively. Thus, the syntax of mercurial signatures used in this work is given by:

- $\mathsf{Setup}(1^\lambda, 1^\ell) \rightarrow (\mathsf{pp}, td)$: Outputs public parameters pp, including parameterized equivalence relations for the message, public key, and secret key spaces: $\mathcal{R}_M$, $\mathcal{R}_{\mathsf{pk}}$, $\mathcal{R}_{\mathsf{sk}}$ and the sample space for key and message converters. This function also outputs a trapdoor (td) that can be used (in conjunction with the corresponding secret key) to recognize public keys.
- $\mathsf{KeyGen}(\mathsf{pp}) \rightarrow (\mathsf{pk}, \mathsf{sk})$: Generates a key pair.
- $\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, M) \rightarrow \sigma$: Signs a message M with the given secret key.
- $\mathsf{Verify}(\mathsf{pp}, \mathsf{pk}, M, \sigma) \rightarrow (0 \text{ or } 1)$: Returns 1 iff σ is a valid signature for M w.r.t. pk.
- $\mathsf{ConvertPK}(\mathsf{pp}, \mathsf{pk}, \rho) \rightarrow \mathsf{pk}'$: Given a key converter ρ, returns pk' by randomizing pk with ρ.
- $\mathsf{ConvertSK}(\mathsf{pp}, \mathsf{sk}, \rho) \rightarrow \mathsf{sk}'$: Randomize a secret key such that it now corresponds to a public key which has been randomized with the same ρ (i.e. signatures from $\mathsf{sk}' = \mathsf{ConvertSK}(\mathsf{pp}, \mathsf{sk}, \rho)$ verify by the randomized $\mathsf{pk}' = \mathsf{ConvertPK}(\mathsf{pk}, \rho)$).
- $\mathsf{ConvertSig}(\mathsf{pp}, \mathsf{pk}, M, \sigma, \rho) \rightarrow \sigma'$: Randomize the signature so that it verifies with a randomized pk' (which has been randomized with the same ρ) and M, but σ' is otherwise unlinkable to σ.
- $\mathsf{ChangeRep}(\mathsf{pp}, \mathsf{pk}, M, \sigma, \mu) \rightarrow (M', \sigma')$: Randomize the message-signature pair such that $\mathsf{Verify}(\mathsf{pk}, M', \sigma') = 1$ (i.e., σ' and σ are indistinguishable) where M' is a new representation of the message equivalence class defined by M.
- $\mathsf{VerifyKey}(\mathsf{pp}, \mathsf{pk}) \rightarrow \{0, 1\}$: Takes a public key and verifies if it is well-formed w.r.t the public parameters pp.
- $\mathsf{VerifyMsg}(\mathsf{pp}, M) \rightarrow \{0, 1\}$: Takes a message and verifies if it is well-formed w.r.t the public parameters pp.

Along with defining the functions above, a mercurial signature construction also defines the equivalence classes that are used in the correctness and security definitions. In the construction of [17], relations and equivalence classes for messages, public keys, and secret key are defined as follows:

$$\mathcal{R}_M = \{(M, M') \in (\mathbb{G}_1^*)^\ell \times (\mathbb{G}_1^*)^\ell | \exists r \in \mathbb{Z}_p^* \text{ s.t. } M' = M^r\}$$

$$\mathcal{R}_{\mathsf{pk}} = \{(\mathsf{pk}, \mathsf{pk}') \in (\mathbb{G}_1^*)^\ell \times (\mathbb{G}_1^*)^\ell | \exists r \in \mathbb{Z}_p^* \text{ s.t. } \mathsf{pk}' = \mathsf{pk}^r\}$$

$$\mathcal{R}_{\mathsf{sk}} = \{(\mathsf{sk}, \mathsf{sk}') \in (\mathbb{Z}_p^*)^\ell \times (\mathbb{Z}_p^*)^\ell | \exists r \in \mathbb{Z}_p^* \text{ s.t. } \mathsf{sk}' = r \cdot \mathsf{sk}\}$$

Equivalence classes are denoted as $[M]_{\mathcal{R}_M}$, $[\mathsf{pk}]_{\mathcal{R}_{\mathsf{pk}}}$, $[\mathsf{sk}]_{\mathcal{R}_{\mathsf{sk}}}$ for messages, public keys, and secret keys respectively, such that: $[M]_{\mathcal{R}_M} = \{M' : (M, M') \in \mathcal{R}_M\}$, $[\mathsf{pk}]_{\mathcal{R}_{\mathsf{pk}}} = \{\mathsf{pk}' : (\mathsf{pk}, \mathsf{pk}') \in \mathcal{R}_{\mathsf{pk}}\}$, $[\mathsf{sk}]_{\mathcal{R}_{\mathsf{sk}}} = \{\mathsf{sk}' : (\mathsf{sk}, \mathsf{sk}') \in \mathcal{R}_{\mathsf{sk}}\}$. Effectively, this means that two messages (M, M') are in the same equivalence class if there exists a randomizer, $\mu \in \mathcal{MC}$, such that $M' = M^{\mu}$ with a similar definition for public keys and secret keys. Because of the properties of equivalence classes (reflexivity, symmetry, and transitivity), the following relations hold: $[M]_{\mathcal{R}_M} = [M']_{\mathcal{R}_M}$ iff $(M, M') \in \mathcal{R}_M$, $[\mathsf{pk}]_{\mathcal{R}_{\mathsf{pk}}} = [\mathsf{pk}']_{\mathcal{R}_{\mathsf{pk}}}$ iff $(\mathsf{pk}, \mathsf{pk}') \in \mathcal{R}_{\mathsf{pk}}$, and $[\mathsf{sk}]_{\mathcal{R}_{\mathsf{sk}}} = [\mathsf{sk}']_{\mathcal{R}_{\mathsf{sk}}}$ iff $(\mathsf{sk}, \mathsf{sk}') \in \mathcal{R}_{\mathsf{sk}}$.

Besides the usual notions for correctness and unforgeability, security of mercurial signatures requires message class-hiding, origin-hiding and public key class-hiding. We recall the original definitions and provide some intuition.

Definition 1 (Correctness [17]**).** *A mercurial signature for parameterized equivalence relations, $\mathcal{R}_{\mathcal{M}}$, $\mathcal{R}_{\mathsf{pk}}$, $\mathcal{R}_{\mathsf{sk}}$, message randomizer space,* sample_{μ}, *and key randomizer space,* sample_{ρ}, *is correct if for all parameters (λ, ℓ),* $\forall(\mathsf{pp}, td) \in \mathsf{Setup}(1^{\lambda}, 1^{\ell})$, *and* $\forall(\mathsf{sk}, \mathsf{pk}) \in \mathsf{KGen}(1^{\lambda})$, *the following holds:*

- ***Verification.*** $\forall M \in \mathcal{M}, \sigma \in \mathsf{Sign}(\mathsf{sk}, M) : \mathsf{Verify}(\mathsf{pk}, M, \sigma) = 1 \wedge \mathsf{VerifyMsg}(\mathsf{pp}, M) = 1 \wedge \mathsf{VerifyKey}(\mathsf{pp}, \mathsf{pk}) = 1$.
- ***Key conversion.*** $\forall \rho \in \mathsf{sample}_{\rho}, (\mathsf{ConvertPK}(\mathsf{pk}, \rho), \mathsf{ConvertSK}(\mathsf{sk}, \rho)) \in \mathsf{KGen}(1^{\lambda})$, $\mathsf{ConvertSK}(\mathsf{sk}, \rho) \in [\mathsf{sk}]_{\mathcal{R}_{\mathsf{sk}}}$, *and* $\mathsf{ConvertPK}(\mathsf{pk}, \rho) \in [\mathsf{pk}]_{\mathcal{R}_{\mathsf{pk}}}$.
- ***Signature conversion.*** $\forall M \in \mathcal{M}, \sigma, \rho \in \mathsf{sample}_{\rho}, \sigma', \mathsf{pk}'$ *s.t* $\mathsf{Verify}(\mathsf{pk}, M, \sigma) = 1$, $\sigma' = \mathsf{ConvertSig}(\mathsf{pk}, M, \sigma, \rho)$, *and* $\mathsf{pk}' = \mathsf{ConvertPK}(\mathsf{pk}, \rho)$, *then* $\mathsf{Verify}(\mathsf{pk}', M, \sigma') = 1$.
- ***Change of message representation.*** $\forall M \in \mathcal{M}, \sigma, \mu \in \mathsf{sample}_{\mu}, M', \sigma'$ *such that* $\mathsf{Verify}(\mathsf{pk}, M, \sigma) = 1$ *and* $(M', \sigma') = \mathsf{ChangeRep}(\mathsf{pk}, M, \sigma; \mu)$ *then* $\mathsf{Verify}(\mathsf{pk}, M', \sigma') = 1$ *and* $M' \in [M]_{\mathcal{R}_M}$.

Unforgeability rules out forgeries on the same equivalence class for messages that have been queried to the signing oracle and public keys as these "forgeries" are actually guaranteed to be computable by correctness. Thus, only forgeries on new equivalence classes are accepted as valid forgeries.

Definition 2 (Unforgeability [17]**).** *A mercurial signature scheme for parameterized equivalence relations $\mathcal{R}_{\mathcal{M}}$, $\mathcal{R}_{\mathsf{pk}}$, $\mathcal{R}_{\mathsf{sk}}$, is unforgeable if for all parameters (λ, ℓ) and all PPT adversaries $\mathcal{A}$, having access to a signing oracle, there exists a negligible function negl such that:*

$$\Pr\left[\begin{array}{r|r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{\ell}) & \mathsf{Verify}(pk^*, M^*, \sigma^*) = 1 \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}) & \wedge\ [pk^*]_{\mathcal{R}_{\mathsf{pk}}} = [pk]_{\mathcal{R}_{\mathsf{pk}}} \\ (\mathsf{pk}^*, M^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}(sk,\cdot)}(\mathsf{pk}) & \wedge\ \forall M \in Q, [M^*]_{\mathcal{R}_M} \neq [M]_{\mathcal{R}_M} \end{array}\right] \leq negl(\lambda)$$

Where Q is the list of messages that the adversary queried to the Sign *oracle.*

Message class-hiding provides unlinkability in the message space, and it's implied by the DDH assumption in the original scheme from [17].

Definition 3 (Message class-hiding [17]). *For all λ, ℓ and all PPT adversaries $\mathcal{A}$, there exists a negligible function negl such that:*

$$\Pr\left[\begin{array}{r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell) \\ M_1 \leftarrow \mathcal{M}; M_2^0 \leftarrow \mathcal{M}; M_2^1 \leftarrow [M_1]_{\mathcal{R}_M} \\ b \leftarrow\$ \{0,1\}; b' \leftarrow \mathcal{A}(\mathsf{pp}, M_1, M_2^b) \end{array} \middle| \, b' = b \right] \leq \frac{1}{2} + negl(\lambda)$$

Importantly, converted signatures should look like freshly computed signatures in the space of all valid ones. This notion is captured with the *origin-hiding* definitions as shown below.

Definition 4 (Origin-Hiding for $\mathsf{ConvertSig}$ [17]). *A mercurial signature scheme is origin-hiding for* $\mathsf{ConvertSig}$ *if, given any tuple* (pk, σ, M) *that verifies, and given a random key randomizer* ρ*,* $\mathsf{ConvertSig}(\sigma, \mathsf{pk}, \rho)$ *outputs a new signature* σ' *such that* σ' *is a uniformly sampled signature in the set of verifying signatures,* $\{\sigma^* | \mathsf{Verify}(\mathsf{ConvertPK}(\mathsf{pk}, \rho), M, \sigma^*) = 1\}$*.*

Definition 5 (Origin-Hiding for $\mathsf{ChangeRep}$ [17]). *A mercurial signature scheme is origin-hiding for* $\mathsf{ChangeRep}$ *if, given any tuple* (pk, σ, M) *that verifies, and given a random message randomizer* μ*,* $\mathsf{ChangeRep}(\mathsf{pk}, M, \sigma, \mu)$ *outputs a new message and signature* M', σ' *such that* M' *is a uniform sampled message in the equivalence class of* M*,* $[M]_{\mathcal{R}_M}$*, and* σ' *is uniformly sampled verifying signature in the set of verifying signatures for* M'*,* $\{\sigma^* | \mathsf{Verify}(\mathsf{pk}, M', \sigma^*) = 1\}$*.*

For anonymous credentials such as the attribute-based credential (ABC) scheme from [20], the notion of message class-hiding is sufficient to provide unlinkability alongside origin-hiding. This is because in ACs the adversary doesn't know the Diffie-Hellman coefficients of the message vector that is signed to produce a credential (these coefficients are only known to the honest user who created the message). In DAC's schemes from mercurial signatures the messages to be signed are public keys, which may be provided by the adversary. Since the adversary knows the corresponding secret key, achieving a class-hiding notion for public keys is much harder. The definition below only considers honestly generated keys for which the adversary doesn't know the secret key. This is similar to the case of message-class hiding but it clearly restricts applications.

Definition 6 (Public key class-hiding [17]).*For all λ, ℓ and all PPT adversaries $\mathcal{A}$, there exists a negligible function (negl) such that:*

$$\Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell); (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{KGen}(\mathsf{pp}); \\ (\mathsf{pk}_2^0, sk_2^0) \leftarrow \mathsf{KGen}(\mathsf{pp}, \ell(\lambda)); \rho \leftarrow\$ (\mathsf{pp}); \\ \mathsf{pk}_2^1 = \mathsf{ConvertPK}(\mathsf{pk}_1, \rho); \mathsf{sk}_2^1 = \mathsf{ConvertSK}(\mathsf{sk}_1, \rho); \\ b \leftarrow\$ \{0,1\}; b' \leftarrow \mathcal{A}^{\mathsf{Sign}(sk_1, \cdot), \mathsf{Sign}(sk_2^b, \cdot)}(\mathsf{pp}, \mathsf{pk}_1, \mathsf{pk}_2^b) \end{array} \middle| \, b' = b \right] \leq \frac{1}{2} + negl(\lambda)$$

Mercurial Signature Construction CL19 [17]**.** We review the mercurial signature construction from [17] in Fig. 1, so that the differences are clear when we present our construction in Sect. 3.

$\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{pp})$: Generate a description of a cryptographic bilinear pairing, BP. Output $\mathsf{pp} = BP$.
$\mathsf{KGen}(\mathsf{pp})$: Sample $\mathsf{sk} = \{x_i\}_{i\in[\ell]} \leftarrow\!\!\$\ \mathbb{Z}_p$ and let $\mathsf{pk} = \{\hat{X}_i\}_{i\in[\ell]}$ where $\forall i \in [\ell], \hat{X}_i = \hat{P}_i^{x_i}$.
$\mathsf{Sign}(\mathsf{sk}, M) \to \sigma$: Sample $y \leftarrow_\$ \mathbb{Z}_p^*$ and compute a signature: $\sigma = \left(Z = (\prod_{i=1}^{\ell} M_i^{x_i})^y, Y = P^{1/y}, \hat{Y} = \hat{P}^{1/y}\right)$.
$\mathsf{Verify}(\mathsf{pk}, M, \sigma) \to (0 \text{ or } 1)$: Accept iff $\prod_{i=1}^{\ell} e(M_i, \hat{X}_i) = e(Z, \hat{Y})$, and $e(Y, \hat{P}) = e(P, \hat{Y})$.
$\mathsf{ConvertSig}(\mathsf{pk}, M, \sigma, \rho) \to \sigma'$: Sample $\psi \leftarrow\!\!\$\ \mathbb{Z}_p$. Compute: $Z' = Z^{\psi\rho}$, $Y' = Y^{1/\psi}$, and $\hat{Y}' = \hat{Y}^{1/\psi}$. Output $\sigma' = (Z', Y', \hat{Y}')$.
$\mathsf{ConvertPK}(\mathsf{pk}, \rho) \to \mathsf{pk}'$: Compute: $\mathsf{pk}' = \mathsf{pk}^\rho$.
$\mathsf{ConvertSK}(\mathsf{sk}, \rho) \to \mathsf{sk}'$: Compute: $\mathsf{sk}' = \rho\mathsf{sk}$.
$\mathsf{ChangeRep}(M, \sigma, \mu) \to (M', \sigma')$: Sample $\psi \leftarrow\!\!\$\ \mathbb{Z}_p$ and compute: $\sigma' = (Z' = Z^{\psi\mu}, Y' = Y^{1/\psi}\hat{Y}' = \hat{Y}^{1/\psi})$, valid for $M' = M^\mu$.

Fig. 1. CL19 Mercurial Signature Construction [17]

We note that a function (RecognizePK shown in Def. 7) can be added to this scheme to recognize any randomization of a public key given a secret key [21]. We use this in our DAC construction to tell if users have been revoked.

Definition 7 (Recognize function for CL19 public keys [21]**).**

– $\mathsf{RecognizePK}(\mathsf{pp}, \mathsf{sk}, \mathsf{pk}) \to \{0, 1\}$ *Parse* pk *as* $\mathsf{pk} = \{\hat{X}_i\}_{i\in[\ell]}$. *Parse* $\mathsf{sk} = \{x_i\}_{i\in[\ell]}$. *Check if* $\forall i \in [\ell - 1], \hat{X}_i^{x_{i+1}/x_i} = \hat{X}_{i+1}$.

3 New Mercurial Signature Construction

In this section we present our core mercurial signature scheme satisfying a stronger notion of adversarial public key class-hiding (APKCH), which will then build the basis for our DAC construction with strong privacy.

3.1 Modified Security Definitions

Subsequently, we present security definitions for our mercurial signature scheme that are modified (or added) when compared to previous work and before going into details we discuss their generality.

On the Generality of our Definitions. Since our main focus in this work is the construction of DAC, we include in our basic definitons (adversarial public-key class hiding and unforgeability) a "levels" parameter L, which tells the challenger how many correlated schemes to construct (i.e., how many levels there will be in the delegation chain of the DAC). In our definitions, after receiving the public parameters for every level, the adversary picks a level, i, and the challenger generates a public key for that level to complete the game with. This allows the reductions in our proofs to ensure that the DAC scheme appears correct to the adversary while reducing APKCH to the anonymity of the DAC scheme. To reduce to unforgeability, we need a similarly modified definition for unforgeability where the challenger generates a number of levels and the adversary chooses which level to continue the game with. Clearly, by setting $L = i = 1$ we obtain versions of the definitions for a standalone mercurial signature scheme. In our definitions, Setup outputs a set of correlated parameters of different signature schemes (pp) and we use pp_i to refer to the parameters that define level i.

First, we formalize the APKCH notion in Def. 8. In this definition, first the challenger generates the public parameters and a public key and gives these to the adversary. The adversary then constructs a message, a key pair and a signature and returns these to the challenger. The challenger then either randomizes the adversary's public key and signature or creates a new signature on the same message from a randomization of the challenger's key. The randomizers are drawn from the "key converter space", $\mathcal{KC}$, which is defined by the construction (commonly, $\mathbb{Z}_p^*$). The adversary is then challenged to determine if the returned key/signature pair is randomized from their own key/signature that they supplied, or if it is a signature from the randomized challenger key. Looking ahead, this property ensures that in a DAC scheme, an adversary cannot determine if they themselves provided a credential to the prover (user) or if another issuer created the credential.

Definition 8 (Adversarial public key class-hiding). *A mercurial signature, Γ, has adversarial public key class-hiding if for all parameters (λ, ℓ, L), the advantage of any PPT set of algorithms $\mathcal{A} = \{\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2\}$, (labeled as $\mathsf{Adv}_{\Gamma,\mathcal{A}}^{\mathsf{APKCH}}(\lambda)$) is negligible,*

$$\mathsf{Adv}_{\Gamma,\mathcal{A}}^{\mathsf{APKCH}}(\lambda) := \left|\Pr\left[\boldsymbol{Exp}_{\Gamma,\mathcal{A}}^{\mathsf{APKCH},0}(\lambda) = 1\right] - \Pr\left[\boldsymbol{Exp}_{\Gamma,\mathcal{A}}^{\mathsf{APKCH},1}(\lambda) = 1\right]\right|$$

where $\boldsymbol{Exp}_{\Gamma,\mathcal{A}}^{\mathsf{APKCH},b}(\lambda)$ is the experiment shown in Fig. 2.

```
1: $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^L)$
2: $(i, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pp})$
3: $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{pp}_i)$;
4: $(\mathsf{pk}_\mathcal{A}, \sigma_\mathcal{A}, M, \mathsf{st}) \leftarrow \mathcal{A}_1^{\mathsf{Sign}(\mathsf{pp}_i, \mathsf{sk}, \cdot)}(\mathsf{pp}_i, \mathsf{pk}, \mathsf{st})$
5: $\sigma \leftarrow \mathsf{Sign}(\mathsf{pp}_i, \mathsf{sk}, M)$
6: $\rho_0 \leftarrow\$ \mathcal{KC}; \mathsf{pk}^0 \leftarrow \mathsf{ConvertPK}(\mathsf{pp}_i, \mathsf{pk}, \rho_0); \sigma^0 \leftarrow \mathsf{ConvertSig}(\mathsf{pp}_i, \sigma, \rho_0)$
7: $\rho_1 \leftarrow\$ \mathcal{KC}; \mathsf{pk}^1 \leftarrow \mathsf{ConvertPK}(\mathsf{pp}_i, \mathsf{pk}_\mathcal{A}, \rho_1); \sigma^1 \leftarrow \mathsf{ConvertSig}(\mathsf{pp}_i, \sigma_\mathcal{A}, \rho_1)$
8: if $\mathsf{Verify}(\mathsf{pp}_i, \mathsf{pk}_\mathcal{A}, \sigma_\mathcal{A}, M) = 1 \ \wedge \ \mathsf{VerifyMsg}(\mathsf{pp}_i, \mathsf{pp}, M) = 1 \ \wedge$
9:   $\mathsf{VerifyKey}(\mathsf{pp}_i, \mathsf{pp}, \mathsf{pk}_\mathcal{A}) = 1$, return $\mathcal{A}_2^{\mathsf{Sign}(\mathsf{pp}_i, \mathsf{sk}, \cdot)}(\mathsf{pp}_i, \mathsf{pp}, \mathsf{st}, \mathsf{pk}^b, \sigma^b)$
10: else return $\mathcal{A}_2^{\mathsf{Sign}(\mathsf{pp}_i, \mathsf{sk}, \cdot)}(\mathsf{pp}_i, \mathsf{pp}, \mathsf{st}, \bot, \bot)$
```

Fig. 2. Adversarial public key class-hiding experiment $\mathbf{Exp}_{\Gamma,\mathcal{A}}^{\mathsf{APKCH},b}(\lambda)$.

Finally, we provide the unforgeability definition that also considers multiple levels in Def. 9.

Definition 9 (Unforgeability). *A mercurial signature scheme for parameterized equivalence relations $\mathcal{R}_{\mathcal{M}}$, $\mathcal{R}_{\mathsf{pk}}$, $\mathcal{R}_{\mathsf{sk}}$, is unforgeable if for all parameters (λ, ℓ, L) and all PPT adversaries $\mathcal{A}$, having access to a signing oracle, there exists a negligible function negl such that:*

$$\Pr\left[\begin{array}{r|r} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell, 1^L) & \\ (i, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pp}) & \mathsf{Verify}(\mathsf{pp}_i, pk^*, M^*, \sigma^*) = 1 \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{pp}_i); & \wedge\ [pk^*]_{\mathcal{R}_{\mathsf{pk}}} = [pk]_{\mathcal{R}_{\mathsf{pk}}} \\ (\mathsf{pk}^*, M^*, \sigma^*) \leftarrow \mathcal{A}_1^{\mathsf{Sign}(\mathsf{pp}_i, sk, \cdot)}(\mathsf{pk}) & \wedge\ \forall M \in Q, [M^*]_{\mathcal{R}_M} \neq [M]_{\mathcal{R}_M} \end{array}\right] \leq negl(\lambda)$$

Where Q is the list of messages that the adversary queried to the Sign *oracle.*

3.2 Construction

In Fig. 3, we construct a mercurial signature (MS) scheme which in particular provides adversarial public key class-hiding (APKCH) as defined in Def. 8. We fix $L = 1$ for this construction and explain how we can set correlated parameters while still achieving APKCH in Sect. 3.3.

As in prior MS constructions, our equivalence classes will be parameterized by the public parameters consisting of a description of the bilinear group $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, P, \hat{P})$. Unlike prior constructions, they will also be parameterized by several length-2ℓ vectors that are part of the public parameters of the system. Specifically, the public parameters will include the vectors $\mathbf{B} = (B_1, \ldots, B_{2\ell})$, $\hat{\mathbf{B}} = (\hat{B}_1, \ldots, \hat{B}_{2\ell})$, and $\hat{\mathbf{V}} = (\hat{V}_1, \ldots, \hat{V}_{2\ell})$, $\mathbf{V} = (V_1, \ldots, V_{2\ell})$. These public parameters have trapdoors which include $\mathbf{b} = (b_1, \ldots, b_\ell)$, $\hat{\mathbf{b}} = (\hat{b}_1, \ldots, \hat{b}_\ell)$, and $\hat{\mathbf{d}} = (\hat{d}_1, \ldots, \hat{d}_\ell)$ which are vectors in $\mathbb{Z}_p^\ell$ and sampled uniformly randomly by Setup. The vector $\mathbf{B}$ will be used to define the message space (and construct messages), while the vector $\hat{\mathbf{B}}$ defines the public key space and allows users to create valid public keys. These public parameters are structured based on the trapdoors, such that $\forall i \in [\ell], B_i = P^{b_i}$, $B_{\ell+i} = B_i^{\hat{b}_i}$, $\forall i \in [\ell], \hat{B}_i = \hat{P}^{\hat{b}_i}$ and $\hat{B}_{\ell+i} = \hat{B}_i^{\hat{d}_i}$. To verify that keys and messages are computed over these bases, we include the verification bases (shown above as $\hat{\mathbf{V}}$ and $\mathbf{V}$) in the public parameters which are constructed as: $\forall i \in [\ell], \hat{V}_i = \hat{P}^{\hat{v}_i b_i}$, $\hat{V}_{\ell+i} = \hat{P}^{\hat{v}_i}$, $\forall i \in [\ell], V_i = P^{v_i \hat{d}_i}$ and $V_{\ell+i} = P^{v_i}$. The vector $\hat{\mathbf{V}}$ is used to verify messages while the vector $\mathbf{V}$ is used to verify keys as described in Sect. 1.3. Structuring the parameters in this way ensures that our construction achieves adversarial public key class-hiding as discussed in Sect. 1.3 and defined in Def. 8. Let pp denote the public parameters.

Our message space will consist of vectors of 2ℓ dimensions over $\mathbb{G}_1$ that have certain structure determined by pp; i.e., not every 2ℓ-dimensional vector will be a valid message. Specifically, our message space, $\mathcal{M}^{pp,\ell}$ is defined as:

$$\mathcal{M}^{pp,\ell} = \{(M_1, \ldots, M_{2\ell}) \mid \exists \mathbf{m} = (m_1, \ldots, m_\ell) \in \mathbb{Z}_p^\ell \text{ such that } \forall 1 \leq i \leq \ell \;\; M_i = B_i{}^{m_i} \wedge M_{\ell+i} = B_{\ell+i}{}^{m_i}\}.$$

Note that, using pp, it is possible to verify that a 2ℓ-dimensional vector is in the message space. In our scheme, the public parameters will include extra bases $\hat{\mathbf{V}} = \{\hat{V}_1, \dots, \hat{V}_{2\ell}\}$ to pair with messages to verify them, i.e. ensuring that $e(M_i, \hat{V}_i) = e(M_{i+\ell}, \hat{V}_{i+\ell})$). Moreover, they include extra bases $\mathbf{V} = \{V_1, \dots, V_{2\ell}\}$, to pair with public keys in the same manner, i.e. $e(V_i, \hat{X}_i) = e(V_{i+\ell}, \hat{X}_{i+\ell})$. We label messages as $\mathbf{M} = \{M_1, \dots, M_{2\ell}\}$ and public keys as $\mathsf{pk} = \{\hat{X}_1, \dots, \hat{X}_{2\ell}\}$. We are now ready to define our equivalence class over the message space, which is the same as in prior work [17]:

$$R_{\mathcal{M}}^{pp,\ell} = \{(\mathbf{M}, \mathbf{M}') : \exists \mu \in \mathbb{Z}_p \text{ s.t. } \mathbf{M}, \mathbf{M}' \in \mathcal{M}^{pp,\ell} \wedge \mathbf{M}' = \mathbf{M}^{\mu}\}.$$

Our public key space will be defined similarly to our message space, but defined over vectors $\hat{\mathbf{B}} = (\hat{P}^{\hat{b}_1}, \dots, \hat{P}^{\hat{b}_\ell}, \hat{P}^{\hat{b}_1 \hat{d}_1}, \dots, \hat{P}^{\hat{b}_\ell \hat{d}_\ell})$ included in pp as well. Like messages, our public key space is a strict subset of the space of 2ℓ-dimension vectors in $\mathbb{G}_2^{2\ell}$, defined below:

$$\mathcal{PK}^{pp,\ell} = \{(\hat{X}_1, \dots, \hat{X}_{2\ell}) \mid \exists \mathbf{x} = (x_1, \dots, x_\ell) \in \mathbb{Z}_p^\ell \text{ such that } \\ \forall 1 \le i \le \ell \;\; \hat{X}_i = \hat{B}_i^{x_i} \wedge \hat{X}_{\ell+i} = \hat{B}_{\ell+i}^{x_i}\}.$$

We define our equivalence classes over the public key space (similarly to [17]):

$$R_{\mathcal{PK}}^{pp,\ell} = \{(\mathsf{pk}, \mathsf{pk}') : \exists \rho \in \mathbb{Z}_p \text{ s.t. } \mathsf{pk}, \mathsf{pk}' \in \mathcal{PK}^{pp,\ell} \wedge \mathsf{pk}' = \mathsf{pk}^{\rho}\}.$$

We will see in the construction that the related structure of these messages and public keys (i.e. the fact that they both use the values $\mathbf{b}$ and $\hat{\mathbf{b}}$) ensures that randomized keys and signatures are not linkable even when the adversary holds the secret key $\{x_1, \dots, x_\ell\}$, while also ensuring that signatures still correctly verify.

Our secret key space takes up the entire space of ℓ-dimensional vectors in $\mathcal{SK}^{pp,\ell} = (\mathbb{Z}_p)^\ell$ and is defined identically to [17] as:

$$R_{\mathsf{sk}}^{pp,\ell} = \{(\mathsf{sk}, \mathsf{sk}') : \exists \rho \in \mathbb{Z}_p \text{ s.t. } \mathsf{sk}, \mathsf{sk}' \in \mathsf{sk}^{pp,\ell} \wedge \mathsf{sk}' = \rho \mathsf{sk}\}$$

In the sequel, when clear from the context, we will omit the superscript pp, ℓ. In our construction, the key and message converter spaces are $\mathcal{KC} = \mathbb{Z}_p^*$ and $\mathcal{MC} = \mathbb{Z}_p^*$.

Our Construction. We are now ready to present our MS construction in Fig. 3 which achieves our APKCH definition for $L = 1$. In Sect. 3.3, we will discuss modifications to the public parameter generation, allowing the scheme's extension for constructing DAC. While verifying the message or public key is not required for unforgeability, we include it in the Verify function as it is needed for public key class-hiding and message class-hiding.

Theorem 1 (Correctness). *The mercurial signature construction in Fig. 3 is correct as described Def. 1.*

Theorem 2 (Unforgeability). *The mercurial signature construction in Fig. 3 meets the unforgeability definition in Def. 2 assuming that the mercurial signature construction in [17] is unforgeable in the generic group model.*

$\mathsf{Setup}(1^\lambda, 1^\ell) \to (\mathsf{pp})$: Sample $\{b_i, \hat{b}_i, \hat{d}_i, \hat{v}_i, v_i\}_{i\in[\ell]} \leftarrow\$ \mathbb{Z}_p$ and compute

$$\mathbf{B} = \{B_i\}_{i\in[2\ell]}, \text{ where } \forall i \in [\ell], B_i \leftarrow P^{b_i}, B_{\ell+i} \leftarrow P^{b_i\hat{b}_i}$$

$$\hat{\mathbf{B}} = \{\hat{B}_i\}_{i\in[2\ell]}, \text{ where } \forall i \in [\ell], \hat{B}_i \leftarrow \hat{P}^{\hat{b}_i}, \hat{B}_{\ell+i} \leftarrow \hat{P}^{\hat{b}_i\hat{d}_i}$$

$$\hat{\mathbf{V}} = \{\hat{V}_i\}_{i\in[2\ell]}, \text{ where } \forall i \in [\ell], \hat{V}_i \leftarrow \hat{P}^{\hat{v}_i\hat{b}_i}, \quad \hat{V}_{\ell+i} \leftarrow \hat{P}^{\hat{v}_i}$$

$$\mathbf{V} = \{V_i\}_{i\in[2\ell]}, \text{ where } \forall i \in [\ell], V_i \leftarrow P^{v_i\hat{d}_i}, \quad V_{\ell+i} \leftarrow P^{v_i}$$

Output: $\mathsf{pp} = (\mathbf{B}, \hat{\mathbf{B}}, \hat{\mathbf{V}}, \mathbf{V})$

$\mathsf{KGen}(\mathsf{pp})$: Sample $\mathsf{sk} = \{x_i\}_{i\in[\ell]} \leftarrow\$ \mathbb{Z}_p$ and let $\mathsf{pk} = \{\hat{X}_i\}_{i\in[2\ell]}$ where $\forall i \in [\ell], \hat{X}_i = \hat{B}_i^{x_i}, \hat{X}_{i+\ell} = \hat{B}_{\ell+i}^{x_i}$.

$\mathsf{VerifyKey}(\mathsf{pp}, \mathsf{pk})$: Accept iff $\forall i \in [\ell], e(V_i, \hat{X}_i) = e(V_{i+\ell}, \hat{X}_{i+\ell})$.

$\mathsf{VerifyMsg}(\mathsf{pp}, M)$: Accept iff $\forall i \in [\ell], e(M_i, \hat{V}_i) = e(M_{i+\ell}, \hat{V}_{i+\ell})$.

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}, M) \to \sigma$: If $\mathsf{VerifyMsg}(\mathsf{pp}, M) = 1$, sample $y \leftarrow\$ \mathbb{Z}_p^*$ and compute a signature:

$\sigma = \left(Z = (\prod_{i=1}^{\ell} M_{\ell+i}^{x_i})^y, Y = P^{1/y}, \hat{Y} = \hat{P}^{1/y}\right)$.

$\mathsf{Verify}(\mathsf{pk}, M, \sigma) \to (0 \text{ or } 1)$: Accept iff $\mathsf{VerifyMsg}(\mathsf{pp}, M) = 1$, $\mathsf{VerifyKey}(\mathsf{pp}, \mathsf{pk}) = 1$,

$\prod_{i=1}^{\ell} e(M_i, \hat{X}_i) = e(Z, \hat{Y})$, and $e(Y, \hat{P}) = e(P, \hat{Y})$.

$\mathsf{ConvertSig}(\mathsf{pp}, \mathsf{pk}, M, \sigma, \rho) \to \sigma'$: Sample $\psi \leftarrow\$ \mathbb{Z}_p$. Compute: $Z' = Z^{\psi\rho}$, $Y' = Y^{1/\psi}$ and $\hat{Y}' = \hat{Y}^{1/\psi}$. Output $\sigma' = (Z', Y', \hat{Y}')$.

$\mathsf{ConvertPK}(\mathsf{pp}, \mathsf{pk}, \rho) \to \mathsf{pk}'$: Compute: $\mathsf{pk}' = \mathsf{pk}^\rho$.

$\mathsf{ConvertSK}(\mathsf{pp}, \mathsf{sk}, \rho) \to \mathsf{sk}'$: Compute: $\mathsf{sk}' = \rho\mathsf{sk}$.

$\mathsf{ChangeRep}(\mathsf{pp}, M, \sigma, \mu) \to (M', \sigma')$: Sample $\psi \leftarrow\$ \mathbb{Z}_p$ and compute: $\sigma' = (Z' = Z^{\psi\mu}, Y' = Y^{1/\psi}\hat{Y}' = \hat{Y}^{1/\psi})$, valid for $M' = M^\mu$.

Fig. 3. Our Mercurial Signature Construction.

We prove Theorem 2 by noting that the CL19 construction [17] is unforgeable in the generic group model, and thus, by showing that our construction's unforgeability relies solely on the unforgeability of the construction of CL19, our construction is also unforgeable in the generic group model.

Theorem 3 (APKCH). *The mercurial signature construction in Fig. 3 meets the* APKCH *definition in 8 in the generic group model.*

Theorem 4 (Origin-hiding of signatures). *The mercurial signature construction in Fig. 3 meets the Origin-hiding of signatures definition in Def. 4.*

Theorem 5 (Origin-hiding of ChangeRep). *The mercurial signature construction in Fig. 3 meets the Origin-hiding of* ChangeRep *definition in Def. 5.*

Theorem 6 (Message class-hiding). *The mercurial signature construction in Fig. 3 meets the Message class-hiding definition in Def. 3.*

The proofs of theorems 4, 5, and 6, follow directly from those of CL19, and are thus omitted here. The proofs of unforgeability (Theorem 2) and APKCH (Theorem 3) are provided in the full version of this paper [22].

3.3 Extending Our Construction to Multiple Levels

In a CL-type DAC scheme, we need chains of public keys that can sign each other. In [17], this is achieved by alternating the source groups of the mercurial signature scheme for each level in the chain. For example, to sign public keys in the highest level of the delegation chain L, if the public keys in level L are in source group $\mathbb{G}_2$, then, in the level $L-1$, a scheme with public keys in $\mathbb{G}_1$ will be used to sign the public keys of level L.

This approach works in [17] because the scheme is symmetric, meaning the public parameters are the same whether public keys are in $\mathbb{G}_2$ or $\mathbb{G}_1$. Unfortunately, our scheme is not symmetrical. Looking at the Setup function in Fig. 3, we see that if we split our message bases and public key bases into halves, $\mathbf{B} = \mathbf{B}^{\mathbf{l}} \| \mathbf{B}^{\mathbf{u}}$ and $\hat{\mathbf{B}} = \hat{\mathbf{B}}^{\mathbf{l}} \| \hat{\mathbf{B}}^{\mathbf{u}}$, the second half of the bases for messages includes the trapdoors $\hat{b}_i$, being formed as $\mathbf{B}^{\mathbf{u}} = \{P^{b_i \hat{b}_i}\}_{i \in [\ell]}$. This trapdoor is included in the first half of the bases for public keys ($\hat{\mathbf{B}}^{\mathbf{l}} = \{\hat{P}^{\hat{b}_i}\}_{i \in [\ell]}$). But, the upper half of the bases for public keys includes the trapdoors, $\hat{d}_i$ ($\hat{\mathbf{B}}^{\mathbf{u}} = \{\hat{P}^{\hat{b}_i \hat{d}_i}\}_{i \in [\ell]}$). The trapdoors $\hat{d}_i$ are not seen in the lower bases for messages ($\mathbf{B}^{\mathbf{l}} = \{P^{b_i}\}_{i \in [\ell]}$). Due to this asymmetry, we cannot simply invert the groups to start signing public keys from higher levels. At first glance, it appears we could fix this by setting $\hat{d}_i = b_i$ (thus allowing messages to be used to sign public keys by computing the signatures on the second half of the public key). Unfortunately, this solution would break the APKCH property of our scheme as computing public keys over $\hat{P}^{b_i \hat{b}_i}$ permits a recognition attack using the bases $P^{b_i \hat{b}_i}$. This forces us to choose a more involved solution.

We can solve this problem using the Setup function in Fig. 4. This function produces $L-1$ levels where the message space of each scheme is exactly the public key space of the subsequent scheme (with the equivalence classes matching as well). To better explain this solution (and simplify our proofs) we discuss the notion of "extending" schemes in this rest of this Section.

```
$\mathsf{Setup}(1^\lambda, 1^\ell, 1^L) \to (\mathsf{pp})$: Sample $\{\hat{b}_{i,j}, v_{i,j}\}_{i \in [\ell], j \in [L]} \leftarrow\$\ \mathbb{Z}_p$, $\{\hat{d}_i\}_{i \in [\ell]} \leftarrow\$\ \mathbb{Z}_p$ and compute the following bases:
    $\hat{\mathbf{B}}_0 = \{\hat{B}_i\}_{i \in [2\ell]}$, where $\forall i \in [\ell], \hat{B}_i \leftarrow P^{\hat{b}_{i,0}}, \hat{B}_{\ell+i} \leftarrow P^{\hat{b}_{i,0} \hat{d}_i}$
    $\mathbf{V}_0 = \{V_i\}_{i \in [2\ell]}$, where $\forall i \in [\ell], V_i \leftarrow \hat{P}^{v_{i,0} \hat{d}_i}, V_{\ell+i} \leftarrow \hat{P}^{v_{i,0}}$
    For $j \in [L] \setminus \{0\}$, if $j \bmod 2 = 0$, $G = \hat{P}, G' = P$ and if $j \bmod 2 = 1$, $G = P, G' = \hat{P}$,
        $\hat{\mathbf{B}}_j = \{\hat{B}_i\}_{i \in [2\ell]}$, where $\forall i \in [\ell], \hat{B}_i \leftarrow G^{\hat{b}_{i,j}}, \hat{B}_{\ell+i} \leftarrow G^{\hat{b}_{i,j} \hat{b}_{i,j-1}}$
        $\mathbf{V}_j = \{V_i\}_{i \in [2\ell]}$, where $\forall i \in [\ell], V_i \leftarrow (G')^{v_{i,j} \hat{b}_{i,j-1}}, V_{\ell+i} \leftarrow (G')^{v_{i,j}}$
    For $j \in [L-1]$: $\mathsf{pp}_j = \{\hat{\mathbf{B}}_{j+1}, \hat{\mathbf{B}}_j, \mathbf{V}_{j+1}, \mathbf{V}_j\}$ such that $\hat{\mathbf{B}}_j, \mathbf{V}_j$ are for keys and $\hat{\mathbf{B}}_{j+1}, \mathbf{V}_{j+1}$ are for messages.
    Output $\mathsf{pp} = \{\mathsf{pp}_j\}_{j \in [L-1]}$.
```

Fig. 4. Parameter generation for multiple levels

We will use the terms "lower" and "higher" to refer to different levels of signature scheme that will be used to construct DAC. The root key is at level 0 which is the lowest level of the delegation chain and user's keys are messages in the $L-1$ (highest) level. In order to sign higher-level public keys with lower-level public keys, starting from our construction in Fig. 3, we need to create multiple levels of the signature scheme so that lower level public key bases can be used to sign public keys from higher levels scheme. To do this, we need to create a new scheme (with public keys in the opposite source group, $\mathbb{G}_1$) with similar structure as in our original scheme in Fig. 3. We recall that in this scheme the message bases and public key bases share the $\hat{b}_i$ trapdoor values as described in the above paragraph. This can be imagined as "extending" a scheme to lower levels. When extending a scheme to enable signatures on the public keys, we'll treat $\hat{d}_i$ as this shared value, setting $\hat{b}_i$ for the lower scheme to be equal to $\hat{d}_i$ in the higher scheme (remember, $\hat{d}_i$ is used in the upper half of the public key bases, $\hat{\mathbf{B}}$ in the public parameters). For example, say we have a higher scheme (with bases $\hat{\mathbf{B}} = \{\hat{B}_i\}_{i\in[2\ell]}$) with key pair $(\mathsf{sk}, \mathsf{pk})$, where $\mathsf{sk} = \{x_i\}_{i\in[\ell]}$, $\mathsf{pk} = \{\hat{X}_i\}_{i\in[2\ell]} = \{\hat{B}_i^{x_i}\}_{i\in[\ell]} \| \{\hat{B}_{\ell+i}^{x_i}\}_{i\in[\ell]}$, $\forall i \in [\ell], \hat{B}_i = \hat{P}^{\hat{b}_i}, \hat{B}_{\ell+i} = \hat{P}^{\hat{b}_i\hat{d}_i}$ and $\hat{b}_i$ and $\hat{d}_i$ are randomly sampled as a trapdoor of the public parameters. We can create a lower scheme (with bases $\hat{\mathbf{B}}' = \{(\hat{B}_i')\}_{i\in[2\ell]}$) and key pair $(\mathsf{sk}', \mathsf{pk}')$ where $\mathsf{sk}' = \{x_i'\}_{i\in[\ell]}$, $\mathsf{pk}' = \{X_i'\}_{i\in[2\ell]} = \{(\hat{B}_i')^{x_i'}\}_{i\in[\ell]} \| \{(\hat{B}_{\ell+i}')^{x_i'}\}_{i\in[\ell]}$ and $\forall i \in [\ell], (\hat{B}_i') = P^{\hat{d}_i}, (\hat{B}_{\ell+i}') = P^{\hat{d}_i\hat{d}_i'}$ and $\hat{d}_i'$ is randomly sampled as a trapdoor of the public parameters. We can now use this lower level scheme to sign the keys in the higher level. We can see that if we form signatures as we did in Fig. 3, these signatures still verify. In Fig. 3, (if we swap the source groups) signatures are formed as $\sigma = (Z, Y, \hat{Y})$ where $Z = (\prod_{i\in[\ell]}(\hat{X}_{\ell+i})^{x_i'})^y$, $Y = \hat{P}^{1/y}$, $\hat{Y} = P^{1/y}$ and y is randomly sampled. We see that $e(\hat{Y}, Z) =>^{\sum_{i\in[\ell]} \hat{d}_i\hat{b}_i x_i x_i'} = \prod_{i\in[\ell]} e(\hat{X}_i, X_i')$ which means that this signature verifies. We provide Fig. 5 to make multi-level signature schemes more clear. In Fig. 5, we can see that when the lower level bases of levels 1 and 2 are paired together, they are equal to the pairing of the higher level bases of level 2, i.e. $e(\hat{B}_{2,i}, \hat{B}_{1,i}) = e(P^{\hat{b}_{2,i}}, \hat{P}^{\hat{b}_{1,i}}) = e(P, \hat{P})^{\hat{b}_{2,i}\hat{b}_{1,i}} = e(\hat{B}_{2,\ell+i}, \hat{P})$. This structure is what ensures that our signatures verify (as described in Sect. 1.3). We've pointed out this structure by highlighting $\hat{b}_{1,i}$ in orange and $\hat{b}_{2,i}$ in blue in 5. This relation holds for levels 2 and 3 as well. Note that in 5, the source groups for level 2 are flipped, meaning that the $\hat{B}_{2,i}$ elements are in $\mathbb{G}_1$ and the Z_2 element is in $\mathbb{G}_2$. For clarity, we keep the notation of the generators, P and $\hat{P}$, correct with respect to levels 1 and 3.

In the full version of this paper [22], we introduce the formal definition of extending parameters. This eases the readability of our proof as a generic group model proof for $L-1$ sets of public parameters simultaneously might be difficult to comprehend. Instead, we prove the APKCH security of a scheme for a single level that reveals enough secrets about the parameters so that the scheme can be extended to match the distribution of the parameters in Fig. 4. A simple hybrid arugment can then be used to prove that our multi-level scheme achieves APKCH as described in Def. 8. More specifically, in the proof of APKCH in the full version, we use a Setup function (similar to Fig. 3) that also reveals $\mathbf{D} = \{D_i\}_{i\in[\ell]} = \{P^{\hat{d}_i}\}_{i\in[\ell]}$ such that if the keys for this scheme live in $\mathbb{G}_2$, then D_i is in $\mathbb{G}_1$. This allows a second setup to be run to create a lower level scheme that is compatible with the first scheme by computing: $\forall i \in [\ell], (\hat{B}_i') =$

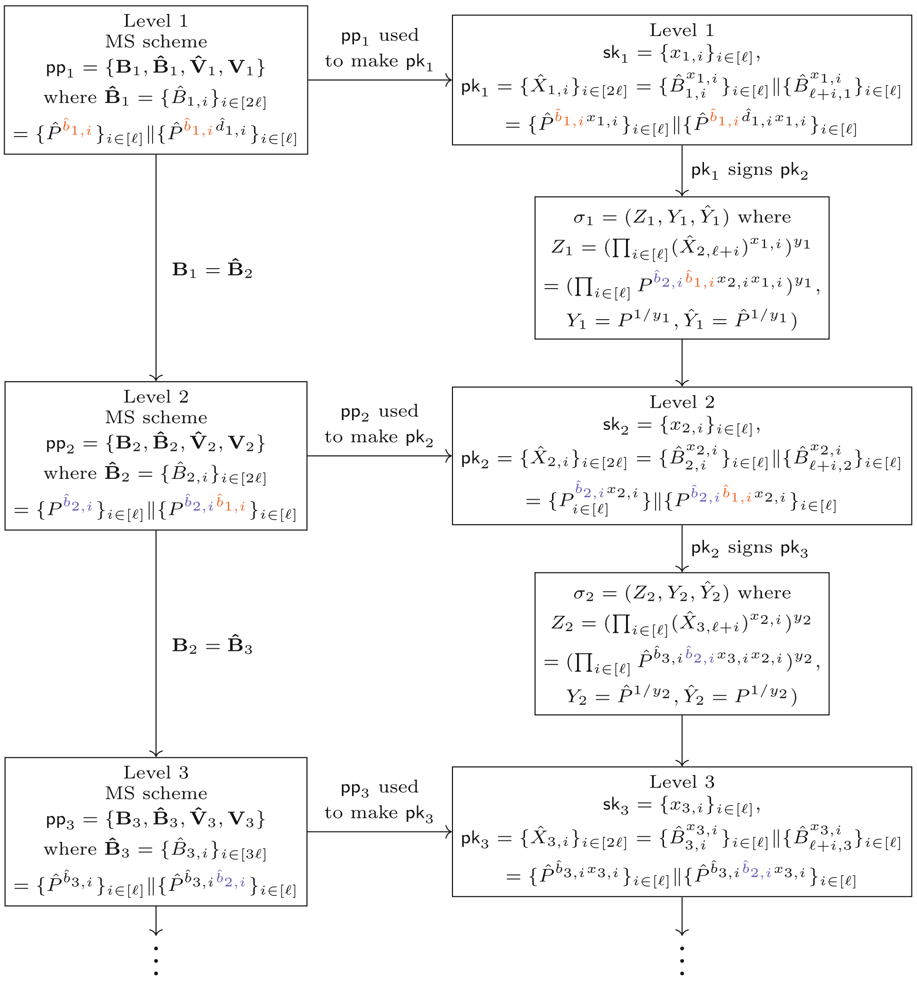

Level 1
MS scheme
$\mathsf{pp}_1 = \{\mathbf{B}_1, \hat{\mathbf{B}}_1, \hat{\mathbf{V}}_1, \mathbf{V}_1\}$
where $\hat{\mathbf{B}}_1 = \{\hat{B}_{1,i}\}_{i\in[2\ell]}$
$= \{\hat{P}^{\hat{b}_{1,i}}\}_{i\in[\ell]} \| \{\hat{P}^{\hat{b}_{1,i}\hat{d}_{1,i}}\}_{i\in[\ell]}$
pp_1 used to make pk_1
Level 1
$\mathsf{sk}_1 = \{x_{1,i}\}_{i\in[\ell]}$,
$\mathsf{pk}_1 = \{\hat{X}_{1,i}\}_{i\in[2\ell]} = \{\hat{B}_{1,i}^{x_{1,i}}\}_{i\in[\ell]} \| \{\hat{B}_{\ell+i,1}^{x_{1,i}}\}_{i\in[\ell]}$
$= \{\hat{P}^{\hat{b}_{1,i}x_{1,i}}\}_{i\in[\ell]} \| \{\hat{P}^{\hat{b}_{1,i}\hat{d}_{1,i}x_{1,i}}\}_{i\in[\ell]}$
pk_1 signs pk_2
$\sigma_1 = (Z_1, Y_1, \hat{Y}_1)$ where
$Z_1 = (\prod_{i\in[\ell]} (\hat{X}_{2,\ell+i})^{x_{1,i}})^{y_1}$
$= (\prod_{i\in[\ell]} P^{\hat{b}_{2,i}\hat{b}_{1,i}x_{2,i}x_{1,i}})^{y_1}$,
$Y_1 = P^{1/y_1}, \hat{Y}_1 = \hat{P}^{1/y_1})$
$\mathbf{B}_1 = \hat{\mathbf{B}}_2$
Level 2
MS scheme
$\mathsf{pp}_2 = \{\mathbf{B}_2, \hat{\mathbf{B}}_2, \hat{\mathbf{V}}_2, \mathbf{V}_2\}$
where $\hat{\mathbf{B}}_2 = \{\hat{B}_{2,i}\}_{i\in[2\ell]}$
$= \{P^{\hat{b}_{2,i}}\}_{i\in[\ell]} \| \{P^{\hat{b}_{2,i}\hat{b}_{1,i}}\}_{i\in[\ell]}$
pp_2 used to make pk_2
Level 2
$\mathsf{sk}_2 = \{x_{2,i}\}_{i\in[\ell]}$,
$\mathsf{pk}_2 = \{\hat{X}_{2,i}\}_{i\in[2\ell]} = \{\hat{B}_{2,i}^{x_{2,i}}\}_{i\in[\ell]} \| \{\hat{B}_{\ell+i,2}^{x_{2,i}}\}_{i\in[\ell]}$
$= \{P_{i\in[\ell]}^{\hat{b}_{2,i}x_{2,i}}\} \| \{P^{\hat{b}_{2,i}\hat{b}_{1,i}x_{2,i}}\}_{i\in[\ell]}$
pk_2 signs pk_3
$\sigma_2 = (Z_2, Y_2, \hat{Y}_2)$ where
$Z_2 = (\prod_{i\in[\ell]} (\hat{X}_{3,\ell+i})^{x_{2,i}})^{y_2}$
$= (\prod_{i\in[\ell]} \hat{P}^{\hat{b}_{3,i}\hat{b}_{2,i}x_{3,i}x_{2,i}})^{y_2}$,
$Y_2 = \hat{P}^{1/y_2}, \hat{Y}_2 = P^{1/y_2})$
$\mathbf{B}_2 = \hat{\mathbf{B}}_3$
Level 3
MS scheme
$\mathsf{pp}_3 = \{\mathbf{B}_3, \hat{\mathbf{B}}_3, \hat{\mathbf{V}}_3, \mathbf{V}_3\}$
where $\hat{\mathbf{B}}_3 = \{\hat{B}_{3,i}\}_{i\in[3\ell]}$
$= \{\hat{P}^{\hat{b}_{3,i}}\}_{i\in[\ell]} \| \{\hat{P}^{\hat{b}_{3,i}\hat{b}_{2,i}}\}_{i\in[\ell]}$
pp_3 used to make pk_3
Level 3
$\mathsf{sk}_3 = \{x_{3,i}\}_{i\in[\ell]}$,
$\mathsf{pk}_3 = \{\hat{X}_{3,i}\}_{i\in[2\ell]} = \{\hat{B}_{3,i}^{x_{3,i}}\}_{i\in[\ell]} \| \{\hat{B}_{\ell+i,3}^{x_{3,i}}\}_{i\in[\ell]}$
$= \{\hat{P}^{\hat{b}_{3,i}x_{3,i}}\}_{i\in[\ell]} \| \{\hat{P}^{\hat{b}_{3,i}\hat{b}_{2,i}x_{3,i}}\}_{i\in[\ell]}$
$\vdots$
$\vdots$

Fig. 5. A series of compatible mercurial signature schemes and a credential chain.

$D_i, (\hat{B}'_{\ell+i}) = D_i^{\hat{d}'_i}$. We can see that this is exactly how the extended scheme computed their public parameters, $(\hat{B}'_{\ell+i})$ (explained earlier in this section) but now the second scheme does not know $\hat{d}_i$, which prevents attacks on class-hiding. Further, because $\mathbf{D}$ lives in $\mathbb{G}_1$ instead of $\mathbb{G}_2$, the adversary cannot use it to create malicious public keys that verify for the original scheme. While the real setup (in Fig. 4) will not reveal $\mathbf{D}$ to an adversary, it is important that the signatures retain their security properties even when this is revealed. Intuitively, this is because schemes built on top of a signature scheme requires some correlated structure. Revealing $D_i = P^{\hat{d}_i}$ in our security games ensures that this correlated structure cannot be leveraged to defeat the security of the schemes at other levels.

4 Delegatable Anonymous Credentials

In this section, we introduce a new DAC construction, showcasing advanced features as strengthened privacy, revocation capabilities, and non-transferability, all while preserving efficiency.

4.1 DAC Functionality

In contrast to non-revocable DAC schemes, our approach integrates a Trusted Revocation Authority (TRA) to efficiently revoke malicious users and maintain a deny list. We outline the high-level functionality of our DAC scheme in Def. 10. This consists of the algorithms: Setup which initializes the scheme, $\mathsf{TKeyGen}$ which generates the TRA's keys, $\mathsf{RootKeyGen}$ which generates the root's keys, $\mathsf{UKeyGen}$ which generates a user's or issuer's keys, $\mathsf{RegisterUser}$ which allows the TRA to distribute revocation tokens, and $\mathsf{RevokeUser}$ which allows the TRA to revoke users. The scheme also consists of the interactive protocols to issue and show credentials: ($\mathsf{Issue} \leftrightarrow \mathsf{Receive}$) and ($\mathsf{Prove} \leftrightarrow \mathsf{Verify}$). This scheme begins with a trusted Setup[2]. Then, the TRA generates an opener secret key and public key using $\mathsf{TKeyGen}$. A root authority (who can be malicious for the sake of anonymity) generates the root key using $\mathsf{RootKeyGen}$ and distributes the root public key to users. A user who wishes to receive a credential runs $\mathsf{UKeyGen}$ and then interacts with the TRA to receive a revocation token by providing their public key to the TRA so that the TRA can run $\mathsf{RegisterUser}$ on it. Subsequently the user interacts with the root (or an issuer that the root has delegated to) using ($\mathsf{Issue} \leftrightarrow \mathsf{Receive}$) and receives a credential (which includes their revocation token). The user then uses their credential and secret key in an interactive protocol ($\mathsf{Prove} \leftrightarrow \mathsf{Verify}$) with any verifier. These verifications can occur at any level within $[L]$ (i.e. for some level, L' such that $L' \leq L$). The verifier can check if the user has been authorized and has not been revoked using the TRA's public key tpk. More specifically, the verifier receives a revocation token for each level in the credential chain from the credential presentation. If the verifier discovers that the prover was malicious through an out-of-band method, they can submit these tokens to the TRA. The TRA will then update their deny list (this deny list is included in the TRA public key for the sake of simplifying the presentation), preventing any future showings that include the user or issuer corresponding to the revocation token from being verified.

We note that our scheme supports a strong model for anonymity where the holder of the root key (colluding with intermediate issuers) cannot de-anonymize users. To model this, we allow the adversary in the anonymity game to choose the root public key along with the corruption of any users of their choice. This allows the adversary to choose any honest user's delegation path from a malicious root with all malicious delegations.

[2] Note that as already discussed in practice this can be done by multiple parties in a sequential way by using ides from updatable common reference strings and only a single party among the set of all parties needs to be trusted.

Definition 10 (Delegatable Anonymous Credentials). *A DAC scheme includes the following algorithms and protocols:*

- $\mathsf{Setup}(1^\lambda, 1^L) \rightarrow (\mathsf{pp}, td)$: Initializes the scheme, outputting public parameters and a trapdoor td.
- $\mathsf{TKeyGen}(\mathsf{pp}) \rightarrow (tsk, tpk)$: Takes pp and outputs a keypair (tsk, tpk) for the TRA. The tpk includes a deny list of revoked users and is continously updated.
- $\mathsf{RootKeyGen}(\mathsf{pp}) \rightarrow (\mathsf{sk}_{rt}, \mathsf{pk}_{rt})$: Generates a key pair used for the root key pk_{rt} (i.e. for level 0) which is trusted for integrity but does not need to be trusted for anonymity.
- $\mathsf{UKeyGen}(\mathsf{pp}, L') \rightarrow (\mathsf{sk}, \mathsf{pk})$: Generates a user's key pair for a specified level.
- $\mathsf{RegisterUser}(\mathsf{pp}, tsk, \mathsf{pk}) \rightarrow (tok)$: Creates a revocation token on the given public key so that the public key can be revoked later with a deny list.
- $(\mathsf{Issue}(\mathsf{sk}_I, \mathsf{cred}_I, L') \leftrightarrow \mathsf{Receive}(\mathsf{sk}_R, tok, L')) \rightarrow \mathsf{cred}_R$: An interactive protocol to receive a signature on a pseudonym. It is run between the two users distinguished by I for issuer or R for receiver. If $L' = 1$ (issuing from the root) then $\mathsf{cred}_I = \perp$.
- $(\mathsf{Prove}(\mathsf{sk}_P, \mathsf{cred}_P, L') \leftrightarrow \mathsf{Verify}(\mathsf{pk}_{rt}, L', tpk)) \rightarrow (b, \{tok_i\}_{i \in [L']})$: A user proves they know a credential on a pseudonym that verifies under the given root key, pk_{rt}. If the verification is successful, the verifier outputs 1 along with a list of revocation tokens for the prover and the chain of credentials. If the verification is unsuccessful, the verifier outputs 0.
- $\mathsf{RevokeUser}(\mathsf{pp}, tsk, tpk, tok) \rightarrow tpk'$: Takes in the TRA's key pair (tsk, tpk) as well as the token for a registered public key and outputs an updated public key tpk' that can be used to recognize any showings in which this public key is part of the chain. For security reasons, this can fail, outputting $\perp$. As an example, we want this function to fail if a malicious tok is provided.

4.2 DAC Security Definitions

In Fig. 6 we formally define the oracles used in our security games. Any formal outputs of oracles are received by the adversary and any modified internal state of the challenger is listed in the description. When calling interactive functions from the DAC scheme (such as $\mathsf{Prove}(\cdot) \leftrightarrow \mathsf{Verify}(\cdot)$), the challenger records the transcript of the interaction in addition to the output of the function. For example, in the $\mathsf{VerifyCred}$ oracle in Fig. 6, we have the challenger interact with the adversary using the Verify function, and in addition to outputting the result of the verification (b) and the list of revocation tokens, $\{tok_i\}_{i \in [L']}$, the protocol also outputs a transcript (τ) of the interaction between the prover and the verifier. Throughout the game, the challenger maintains some state to keep track of honest users and credentials that were given to the adversary. This global state is used in the unforgeability game. Specifically, the challenger keeps track of one set, $\mathsf{DEL}_\mathcal{A}$ to keep track of what has been delegated to the adversary. Moreover, the challenger initializes three maps to keep track of honest user state, SK holds user secret keys, CRED holds user credentials, and LVL records what level a user's credential is for. They are as follows: $\mathsf{SK} : \mathcal{H} \rightarrow \mathcal{SK}$, $\mathsf{CRED} : \mathcal{SK} \rightarrow \mathcal{CRED}$ and

$\mathsf{LVL} : \mathcal{SK} \to [L]$, where $\mathcal{H}$ is the set of all honest user handles (which the adversary uses to refer to honest users), $\mathcal{SK}$ is the set of all secret keys, and $\mathcal{CRED}$ is the set of all credentials.

```
$\mathcal{O}^{\mathsf{CreateHonestUser}}(id_I, id_R, L') \to (\mathsf{pk})$
if $\mathsf{SK}[id_R] \neq \perp$, return $\perp$
if $\mathsf{SK}[id_I] = \perp$, return $\perp$
if $\mathsf{LVL}[\mathsf{SK}[id_I]] \neq L' - 1$, return $\perp$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$
$\mathsf{SK}[id_R] = \mathcal{E}_{\mathsf{pk}}(\mathsf{pk})^{\dagger}$
$\mathsf{LVL}[\mathsf{SK}[id_R]] = L'$
$tok = \mathsf{RegisterUser}(\mathsf{pp}, tsk, \mathsf{pk})$
$(\mathsf{cred}, \tau) \leftarrow (\mathsf{Receive}(\mathsf{pp}, \mathsf{SK}[id_R], tok, \mathsf{pk}_{rt}, L')$
    $\leftrightarrow \mathsf{Issue}(\mathsf{pp}, \mathsf{SK}[id_I], \mathsf{CRED}[id_I], L'))$
$\mathsf{CRED}[id_R] = \mathsf{cred}$
return $(\mathsf{pk})$

$\mathcal{O}^{\mathsf{ReceiveCred}}(id_R, L') \leftrightarrow \mathcal{A}$
if $\mathsf{SK}[id_R] \neq \perp$, return $\perp$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{pp})$
$\mathsf{SK}[id_R] = \mathcal{E}_{\mathsf{pk}}(\mathsf{pk})^{\dagger}$
$\mathsf{LVL}[\mathsf{SK}[id_R]] = L'$
$tok = \mathsf{RegisterUser}(\mathsf{pp}, tsk, \mathsf{pk})$
$(\mathsf{cred}, \tau) \leftarrow (\mathsf{Receive}(\mathsf{pp}, \mathsf{SK}[id_R], tok, \mathsf{pk}_{rt}, L)$
    $\leftrightarrow \mathcal{A}(\mathsf{pk}))$
if $\mathsf{cred} \neq \perp \wedge \forall i \in [L'], (*, i) \notin \mathsf{DEL}_{\mathcal{A}}$,
    $forgery = 1$
$\mathsf{CRED}[id_R] = \mathsf{cred}$
return $\perp$

$\mathcal{O}^{\mathsf{IssueFrom}}(id_I) \leftrightarrow \mathcal{A}$
if $\mathsf{SK}[id_I] = \perp$, return $\perp$
if $\mathsf{CRED}[id_I] = \perp$, return $\perp$
$(\mathsf{cred}, \tau) \leftarrow (\mathsf{Issue}(\mathsf{SK}[id_I], \mathsf{CRED}[id_I])$
    $\leftrightarrow \mathcal{A})$
$\mathsf{DEL}_{\mathcal{A}} = \mathsf{DEL}_{\mathcal{A}} \cup \{(\mathcal{E}_{\mathsf{pk},R}(\tau),$
    $\mathsf{LVL}[\mathsf{SK}[id_I]])\}$
return $\mathsf{cred}$

$\mathcal{O}^{\mathsf{ProveTo}}(id) \leftrightarrow \mathcal{A}$
if $\mathsf{SK}[id] = \perp$, return $\perp$
if $\mathsf{CRED}[id] = \perp$, return $\perp$
$\mathsf{Prove}(\mathsf{pp}, \mathsf{SK}[id], \mathsf{CRED}[id], \mathsf{pk}_{rt})$
    $\leftrightarrow \mathcal{A}$
return $\perp$

$\mathcal{O}^{\mathsf{RegisterUser}}(\mathsf{pk}) \to tok$
return $\mathsf{RegisterUser}(\mathsf{pk})$

$\mathcal{O}^{\mathsf{RevokeUser}}(\mathsf{pk})$
$tpk' = \mathsf{RevokeUser}(\mathsf{pp}, tsk, \mathsf{pk})$
if $tpk' \neq \perp$,
    $tpk' = tpk$
    $\mathsf{SK}_{DL} = \mathsf{SK}_{DL} \cup \{\mathcal{E}_{\mathsf{sk}}(\mathsf{pk})\}$
return $tpk'$
```

Fig. 6. Definition of Oracles. † The oracle uses $\mathcal{E}_{\mathsf{pk}}$ to ensure $\mathsf{SK}[id]$ holds a canonical representation of the secret key.

The root key is included in SK with handle $id = 0$ where $\mathsf{LVL}[\mathsf{SK}[0]] = 0$ and $\mathsf{CRED}[\mathsf{SK}[0]] = \perp$. The challenger also keeps track of what keys have been added to the deny list with the list $\mathsf{SK}_{DL} \subset \mathcal{SK}$. At the start of any game, the challenger initializes all sets to the empty set and initializes all maps to be degenerate, such as mapping $\forall i \in \mathcal{H}, \mathsf{SK}[i] = \perp$. In the unforgeability game, we give the adversary access to all of the oracles. In the $\mathcal{O}^{\mathsf{CreateHonestUser}}$ oracle, the adversary specifies two users with one issuing a credential to the other. We initialize users at the same time that they are issued a credential to simplify the scheme. As an example use of this oracle, the adversary can specify $id_I = 0$ at the start of a game to have a credential be issued from the root. We do not allow the adversary to issue to a user multiple times, and thus if the specified user already exists when the adversary calls $\mathcal{O}^{\mathsf{CreateHonestUser}}$, then the challenger aborts. We allow the adversary to issue credentials to honest users using the $\mathcal{O}^{\mathsf{ReceiveCred}}$ oracle. In this oracle, the adversary specifies a user to receive a credential at a specified level. If the adversary was never issued a credential that would allow them to delegate this credential to the honest user, the challenger set a forgery flag in the global state (labeled *forgery*) which is checked in the unforgeability game. In the $\mathcal{O}^{\mathsf{IssueFrom}}$ oracle, the adversary receives a credential from an honest user and the challenger records which adversarial key received this credential at which level. In the $\mathcal{O}^{\mathsf{ProveTo}}$ oracle, the adversary acts as the verifier for a user. In the $\mathcal{O}^{\mathsf{RegisterUser}}$ oracle the adversary can receive a revocation token for one of their public keys. In the $\mathcal{O}^{\mathsf{RevokeUser}}$ oracle, the adversary can add a user to the deny list.

Anonymity: Our anonymity definition is shown in Def 11. The anonymity game involves the adversary and the challenger. The adversary controls all participants, including the root credential authority (but does not control the TRA). The game proceeds as follows: The challenger generates the public parameters, which are given to the adversary along with the registrar's public key and access to a registration and revocation oracle. The adversary creates two credential chains of the same length and provides the secret keys of the end users of these chains to the challenger. The challenger ensures that they are valid credential chains. The challenger randomly selects one of the users and proves possession of the corresponding credential chain to the adversary. The adversary wins if it can correctly identify which user the challenger picked. No honest users are created in this game, as the adversary controls all aspects except for the registration and revocation oracles. To model issuer privacy and showing privacy, the adversary outputs a bit, j, to indicate whether the challenger should act as the issuer or the shower. We formalize this game in Def. 11.

Definition 11 (Anonymity). *A DAC scheme is anonymous if the advantage any PPT adversary ($\mathcal{A} = \{\mathcal{A}_0, \mathcal{A}_1\}$) in the following anonymity game, defined by the chance that the game outputs* 1*, is* $1/2 + negl(\lambda)$*:*

```
1: $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^L)$
2: $(tsk, tpk) \leftarrow \mathsf{TKeyGen}(\mathsf{pp})$
3: $(, , j, \mathsf{pk}_{rt}, \mathsf{sk}_0, \mathsf{cred}_0, \mathsf{sk}_1, \mathsf{cred}_1, L') \leftarrow \mathcal{A}_0^{\mathcal{O}^{\mathsf{RegisterUser}}(\cdot), \mathcal{O}^{\mathsf{RevokeUser}}(\cdot)}(\mathsf{pp}, tpk)$
4: $\forall i \in \{0,1\}$ :
5:    if $(\mathsf{Prove}(\mathsf{pp}, \mathsf{sk}_i, \mathsf{cred}_i, L') \leftrightarrow \mathsf{Verify}(\mathsf{pk}_{rt}, L', tpk)) \neq (1, *)$, return $\perp$
6: $b \leftarrow_\$ \{0,1\}$
7: if $j = 0, b' \leftarrow (\mathsf{Prove}(\mathsf{pp}, \mathsf{sk}_b, \mathsf{cred}_b, L') \leftrightarrow \mathcal{A}_1(st))$
8: if $j = 1, b' \leftarrow (\mathsf{Issue}(\mathsf{pp}, \mathsf{sk}_b, \mathsf{cred}_b, L' + 1) \leftrightarrow \mathcal{A}_1(st))$
9: return $b' = b$
```

Unlike the anonymity definition in [17], we allow the adversary to participate in the challenge credential chain. Therefore, we do not need to control the state of the game with the challenger; in the anonymity game, the challenger only performs the role of the TRA and the challenge user. In addition, we aim to maintain the anonymity of honest users even when the anonymity of some adversarial users is revoked. This new definition represents a simplified and more comprehensive anonymity model, which we present as a novel contribution.

Unforgeability: Our unforgeability game is simpler than [17], even though it is conceptually similar. We remove oracles that reveal pseudonyms of honest users. Revealing pseudonyms alone has no real-world use-case in DAC and the adversary effectively reveals pseudonyms during a showing anyway. Also, we integrate user creation with credential issuance, as a user's key pair is not used until it is associated with a credential. Otherwise, our unforgeability definition (Def. 12) is mostly unchanged from [17], but we add the RegisterUser and RevokeUser functions that facilitate revocation.

Moreover, our unforgeability definition ensures that the adversary was correctly delegated a credential on line 10 in Def. 12, and that none of the keys in the adversary's credential are on the deny list, on line 11.

To ensure that the challenger can check the key classes, we parameterize the definition with the extractor, $\mathcal{E}_{\mathsf{pk}}$, which takes in a public key and extracts a secret key from it. If $\mathcal{E}_{\mathsf{pk}}$ is run on the transcript of a showing, it extracts the secret key of the credential holder. If $\mathcal{E}_{\mathsf{pk}}$ is run on the transcript of an issuing, it extracts the secret key of the issuer. We denote these by $\mathcal{E}_{\mathsf{pk},R}$ for receiver and $\mathcal{E}_{\mathsf{pk},I}$ for issuer. This extractor must extract the same secret key no matter how the public key has been randomized. For mercurial signatures, this means that the extractor extracts a canonical secret key which is constant over any representation of the equivalence class of secret keys. We also assume an extractor $\mathcal{E}_{\mathsf{cred}}$ that can take in a credential or the transcript of a showing of a credential and output the canonical secret keys used in the delegation chain including the end user of the credential.

Definition 12 (Unforgeability). *A DAC scheme is unforgeable if any PPT adversary's advantage in the following game, defined by the chance that the game outputs* 1*, is negligible in* λ*.* $\mathcal{A}$ *is given all the oracles from Fig 6 labeled as* $\mathcal{O}$*.*

```
1:  $(\mathsf{pp}, td) \leftarrow \mathsf{Setup}(1^\lambda, 1^L)$
2:  $(tsk, tpk) \leftarrow \mathsf{TKeyGen}(\mathsf{pp})$
3:  $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{RootKeyGen}(\mathsf{pp})$
4:  $\mathsf{SK}[0] = \mathsf{sk}; \mathsf{pk}_{rt} = \mathsf{pk}$
5:  $(st, L) \leftarrow \mathcal{A}_0^{\mathcal{O}}(\mathsf{pp}, \mathsf{pk})$
6:  $((b, *), \tau) \leftarrow (\mathsf{Verify}(\mathsf{pk}_{rt}, L, tpk) \leftrightarrow \mathcal{A}_1(st)$
7:  $\{\mathsf{sk}_i\}_{i \in [L']} \leftarrow \mathcal{E}_{\mathsf{cred}}(\tau)$
8:  if $forgery = 1$, return $1$
9:  if $\mathsf{sk}_0 \neq \mathsf{sk}_{rt}$, return $b$
10: if $\forall i \in [L'], (\mathsf{sk}_i, i) \notin \mathsf{DEL}_{\mathcal{A}}$, return $b$
11: if $\exists i \in [L'],\ s.t.\ \mathsf{sk}_i \in \mathsf{SK}_{DL}$, return $b$
12: return $0$
```

4.3 DAC Construction

Our DAC construction uses the multi-level setup function from Fig. 4.

To simplify our DAC construction, we add a function $\mathsf{TracePK}$, which takes in a "linker" and a revocation token and returns if this linker is associated with the revocation token. These linker values will be stored in the deny list in the TRA's public key tpk.

Definition 13 (DAC construction).

- $\mathsf{Setup}(1^\lambda, 1^L) \to (\mathsf{pp}, td)$: Call the setup function described in Fig. 4 which generates L correlated parameters for our signature scheme in Fig. 3, $\{\mathsf{pp}_i\}_{i\in[L]} = \mathsf{MultiSetup}(1^\lambda, 1^{\ell=2}, 1^L)$. Then initialize extra bases for the revocation authority and the root authority using the CL19 scheme, $(\mathsf{pp}_{\mathsf{CL19}}) \leftarrow \mathsf{Setup}_{\mathsf{CL19}}(1^\lambda, 1^{\ell=2})$. Output $\mathsf{pp} = (\{\mathsf{pp}_i\}_{i\in[L]}, \mathsf{pp}_{\mathsf{CL19}})$, $td = (\{td_i\}_{i\in[\ell]})$.
- $\mathsf{RootKeyGen}(\mathsf{pp}) \to (\mathsf{sk}_{rt}, \mathsf{pk}_{rt})$: Generate a CL19 key pair using $\mathsf{pp}_{\mathsf{CL19}}$.
- $\mathsf{UKeyGen}(\mathsf{pp}, L') \to (\mathsf{sk}, \mathsf{pk})$: Create a secret key for the corresponding scheme, $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KGen}(\mathsf{pp}_{L'})$. The user initializes their credential chain as $chain = \perp$.
- $\mathsf{TKeyGen}(\mathsf{pp}) \to (tsk, tpk)$: Create a CL19 key $(\mathsf{sk}, \mathsf{pk})$ of length $\ell = 2$, $(tsk_{\mathsf{sk}}, tpk_{\mathsf{pk}}) \leftarrow \mathsf{KGen}(\mathsf{pp}_{\mathsf{CL19}})$. Initialize a set of linkers, $tsk_{link} = \emptyset$. Initialize a deny list, $DL = \emptyset$. Let $tsk_{\mathsf{sk}} = \mathsf{sk}$, $tsk = (tsk_{\mathsf{sk}}, tsk_{link})$, and $tpk = (tpk_{\mathsf{pk}}, DL)$.
- $\mathsf{RegisterUser}(\mathsf{pp}, tsk, \mathsf{pk}) \to (tok)$: Generate a new key pair $(\mathsf{sk}_{rev}, \mathsf{pk}_{rev})$ using $\mathsf{pp}_{\mathsf{CL19}}$. Sign pk_{rev} using tsk_{sk} where $tsk = (tsk_{\mathsf{sk}}, tsk_{link})$ yielding σ_0. Then, use sk_{rev} to sign pk, yielding σ_1. This yields the revocation token, $tok = (\mathsf{pk}_{rev}, \sigma_0, \sigma_1)$. The secret key, sk_{rev}, will serve as the linker for this revocation token and it is denoted as $link$. Save this linker in the TRA's state, $tsk'_{link} = tsk_{link} \cup \{link\}$ and update the state: $tsk' = (tsk_{\mathsf{sk}}, tsk'_{link})$. Output revocation token tok.
- $\mathsf{RevokeUser}(\mathsf{pp}, tsk, tpk, tok) \to tpk'$: Iterate through the linkers $(link_i)$ in tsk_{link} and check if $\mathsf{TracePK}(\mathsf{pp}, link_i, tok) = 1$ for each of them. If this holds for a linker, $link_i$, concatenate $link_i$ to the linkers in tpk (the deny list) and output this new public key as tpk'.
- $\mathsf{TracePK}(\mathsf{pp}, link, tok) \to \{0, 1\}$: Parse tok as $tok = (\mathsf{pk}_{rev}, \sigma_0, \sigma_1)$. Check if $\mathsf{RecognizePK}(\mathsf{pp}_{\mathsf{CL19}}, link, \mathsf{pk}_{rev}) = 1$ (cf. Section 2), i.e., parse pk_{rev} as $\mathsf{pk}_{rev} = (\hat{X}_1, \hat{X}_2)$. Parse $link = (x_1, x_2)$. Check if $\hat{X}_1^{x_2/x_1} = \hat{X}_2$. If this holds, output 1. Otherwise, output 0.
- $(\mathsf{Issue}(\mathsf{sk}_I, \mathsf{cred}_I, L') \leftrightarrow \mathsf{Receive}(\mathsf{sk}_R, tok, L')) \to \mathsf{cred}_R$: The receiver samples $\rho \leftarrow \mathcal{KC}$ (from the set of key converters) and generates a randomized public key from their secret key, pk', using the randomization factor, ρ. They also randomize their revocation token, tok, yielding, $tok = (\mathsf{pk}'_{rev}, \sigma'_0, \sigma'_1)$, such that $\mathsf{Verify}_{\mathsf{CL19}}(\mathsf{pp}_{\mathsf{CL19}}, tpk, \mathsf{pk}'_{rev}, \sigma'_0) = 1$ and $\mathsf{Verify}_{\mathsf{CL19}}(\mathsf{pp}_{\mathsf{CL19}}, \mathsf{pk}'_{rev}, \mathsf{pk}', \sigma'_0) = 1$. The receiver sends over pk'. The issuer then randomizes all public keys in their credential chain along with the signatures, randomizing their secret key to match. They also randomize all revocation tokens in their chain as described above. They then sign pk' yielding a signature, σ. They send their randomized credential chain, $chain$, along with σ to the receiver. The receiver computes the chain, $chain' = chain \| (\mathsf{pk}', \sigma, tok')$.

The receiver stores their credential as $\mathsf{cred} = (chain', \rho)$. The randomizer is also stored to ensure the receiver can correctly randomize their secret key to match their public key in the chain.

- $(\mathsf{Prove}(\mathsf{sk}_P, \mathsf{cred}_P, L') \leftrightarrow \mathsf{Verify}(\mathsf{pk}_{rt}, L', tpk)) \rightarrow (b, \{tok_i\}_{i \in [L']})$: The prover randomizes all public keys and signatures in their credential cred using $\rho^* = \rho * \rho'$ where ρ is the randomizer found in their credential and ρ' is randomly sampled. They send over their randomized credential chain, *chain*, and perform an interactive proof of knowledge that they know the sk that corresponds to the last public key in the chain. The verifier then verifies each public key with the signatures. The verifier also iterates through the revocation tokens in the credential chain checks that for each public key pk_i and $tok_i = (\mathsf{pk}_{rev,i}, \sigma_{i,0}, \sigma_{i,1})$ in the chain it holds that $\mathsf{Verify}_{\mathsf{CL19}}(tpk, \mathsf{pk}_{rev,i}, \sigma_{i,0}) = 1$ and $\mathsf{Verify}_{\mathsf{CL19}}(\mathsf{pk}_{rev,i}, \mathsf{pk}_i, \sigma_{i,1}) = 1$. They then also iterate through each $link_j \in tpk$ and ensure that $\mathsf{TracePK}(\mathsf{pp}, link_j, tok_i) = 0$ for each level i in the length of the chain. If all these checks hold, the verifier outputs 1 and if any checks fail, the verifier outputs 0. The verifier also outputs all of the tok_i values received from the prover.

Theorem 7 (Correctness of the construction in Def. 13). *Our construction in Def. 13 is correct as defined in the full version of this paper [22].*

Theorem 8 (Unforgeability of the construction in Def. 13). *If the underlying signature scheme is unforgeable with respect to Def. 2, our construction in Def. 13 is unforgeable with respect to Def. 12.*

Theorem 9 (Anonymity of the construction in Def. 13). *If the underlying signature scheme has origin-hiding and has adversarial public key-class hiding, the DAC scheme in Def. 13 is anonymous with respect to definition 11.*

We prove these theorems (and define correctness) in the full version of this paper [22].

5 Conclusion and Future Work

In this paper, we constructed mercurial signatures with stronger security properties than seen in the literature, which could potentially be used for many privacy-preserving schemes just as the first such signatures in [17]. We use it as a basis for an efficient DAC scheme with strong privacy guarantees and delegator revocation functionality. Our DAC construction could be further adapted to support attributes extending its functionality, where the technique from [32] seems promising. We leave this extension to future work.

Acknowledgments. We are very grateful to the anonymous reviewers for their many helpful comments and suggestions. Omid Mir was supported by the European Union's Horizon Europe project SUNRISE (project no. 101073821), and by PREPARED, a project funded by the Austrian security research programme KIRAS of the Federal Ministry of Finance (BMF). Scott Griffy and Anna Lysysanskaya were supported by NSF grants 2247305, 2154170, and 2312241 as well as the Ethereum Foundation.

References

1. Acar, T., Nguyen, L.: Revocation for delegatable anonymous credentials. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 423–440. Springer, Berlin, Heidelberg (Mar 2011). https://doi.org/10.1007/978-3-642-19379-8_26
2. Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Berlin, Heidelberg (Aug 2009).https://doi.org/10.1007/978-3-642-03356-8_7
3. Ben-Sasson, E., Chiesa, A., Green, M., Tromer, E., Virza, M.: Secure sampling of public parameters for succinct zero knowledge proofs. In: 2015 IEEE Symposium on Security and Privacy. pp. 287–304. IEEE Computer Society Press (May 2015).https://doi.org/10.1109/SP.2015.25
4. Blömer, J., Bobolz, J.: Delegatable attribute-based anonymous credentials from dynamically malleable signatures. In: Preneel, B., Vercauteren, F. (eds.) ACNS 18International Conference on Applied Cryptography and Network Security. LNCS, vol. 10892, pp. 221–239. Springer, Cham (Jul 2018). https://doi.org/10.1007/978-3-319-93387-0_12
5. Boneh, D., Shacham, H.: Group signatures with verifier-local revocation. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004. pp. 168–177. ACM Press (Oct 2004).https://doi.org/10.1145/1030083.1030106
6. Bowe, S., Gabizon, A., Green, M.D.: A multi-party protocol for constructing the public parameters of the pinocchio zk-SNARK. In: Zohar, A., Eyal, I., Teague, V., Clark, J., Bracciali, A., Pintore, F., Sala, M. (eds.) FC 2018 Workshops. LNCS, vol. 10958, pp. 64–77. Springer, Berlin, Heidelberg (Mar 2019). https://doi.org/10.1007/978-3-662-58820-8_5
7. Brorsson, J., David, B., Gentile, L., Pagnin, E., Wagner, P.S.: PAPR: Publicly auditable privacy revocation for anonymous credentials. In: Rosulek, M. (ed.) CT-RSA 2023. LNCS, vol. 13871, pp. 163–190. Springer, Cham (Apr 2023).https://doi.org/10.1007/978-3-031-30872-7_7
8. Camenisch, J., Drijvers, M., Dubovitskaya, M.: Practical UC-secure delegatable credentials with attributes and their application to blockchain. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 683–699. ACM Press (Oct / Nov 2017).https://doi.org/10.1145/3133956.3134025
9. Camenisch, J., Kohlweiss, M., Soriente, C.: An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 481–500. Springer, Berlin, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00468-1_27
10. Camenisch, J., Kohlweiss, M., Soriente, C.: Solving revocation with efficient update of anonymous credentials. In: Garay, J.A., Prisco, R.D. (eds.) SCN 10. LNCS, vol. 6280, pp. 454–471. Springer, Berlin, Heidelberg (Sep 2010). https://doi.org/10.1007/978-3-642-15317-4_28
11. Camenisch, J., Krenn, S., Lehmann, A., Mikkelsen, G.L., Neven, G., Pedersen, M.Ø.: Formal treatment of privacy-enhancing credential systems. In: Dunkelman, O., Keliher, L. (eds.) SAC 2015. LNCS, vol. 9566, pp. 3–24. Springer, Cham (Aug 2016).https://doi.org/10.1007/978-3-319-31301-6_1
12. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Berlin, Heidelberg (May 2001).https://doi.org/10.1007/3-540-44987-6_7

13. Camenisch, J., Lysyanskaya, A.: Dynamic accumulators and application to efficient revocation of anonymous credentials. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 61–76. Springer, Berlin, Heidelberg (Aug 2002).https://doi.org/10.1007/3-540-45708-9_5
14. Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 78–96. Springer, Berlin, Heidelberg (Aug 2006).https://doi.org/10.1007/11818175_5
15. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. Commun. ACM **28**(10), 1030–1044 (oct 1985). https://doi.org/10.1145/4372.4373
16. 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, Cham (Mar 2022). https://doi.org/10.1007/978-3-030-97121-2_15
17. 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, Cham (Mar 2019).https://doi.org/10.1007/978-3-030-12612-4_27
18. Crites, E.C., Lysyanskaya, A.: Mercurial signatures for variable-length messages. PoPETs **2021**(4), 441–463 (2021). https://doi.org/10.2478/popets-2021-0079
19. 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, Cham (Dec 2015). https://doi.org/10.1007/978-3-319-27239-9_4
20. 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
21. Griffy, S., Lysyanskaya, A.: PACIFIC. IACR Communications in Cryptology **1**(2) (2024). https://doi.org/10.62056/ay11fhbmo
22. Griffy, S., Lysyanskaya, A., Mir, O., Kempner, O.P., Slamanig, D.: Delegatable anonymous credentials from mercurial signatures with stronger privacy. Cryptology ePrint Archive, Report 2024/1216 (2024), https://eprint.iacr.org/2024/1216
23. Groth, J., Kohlweiss, M., Maller, M., Meiklejohn, S., Miers, I.: Updatable and universal common reference strings with applications to zk-SNARKs. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 698–728. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96878-0_24
24. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Berlin, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_24
25. 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, Berlin, Heidelberg (Dec 2014).https://doi.org/10.1007/978-3-662-45611-8_26
26. 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 (Nov 2021).https://doi.org/10.1145/3460120.3484582
27. 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.) ACM CCS 2023. pp. 30–44. ACM Press (Nov 2023). https://doi.org/10.1145/3576915.3623203

28. Mir, O., Slamanig, D., Bauer, B., Mayrhofer, R.: Practical delegatable anonymous credentials from equivalence class signatures. Proc. Priv. Enhancing Technol. **2023**(3), 488–513 (2023). https://doi.org/10.56553/POPETS-2023-0093
29. Abe, M., Nanri, M., Perez Kempner, O., Tibouchi, M.: Interactive threshold mercurial signatures and applications. Cryptology ePrint Archive, Paper 2024/625 (2024). https://doi.org/10.1007/978-981-96-0891-1_3, https://eprint.iacr.org/2024/625
30. 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 - 22nd International Conference, ACNS 2024, Abu Dhabi, United Arab Emirates, March 5-8, 2024, Proceedings, Part III. Lecture Notes in Computer Science, vol. 14585, pp. 105–134. Springer (2024). https://doi.org/10.1007/978-3-031-54776-8_5
31. Putman, C., Martin, K.M.: Selective delegation of attributes in mercurial signature credentials. Cryptology ePrint Archive, Report 2023/1896 (2023), https://eprint.iacr.org/2023/1896
32. 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)
33. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Proceedings of the 16th Annual International Conference on Theory and Application of Cryptographic Techniques. p. 256–266. EUROCRYPT'97, Springer-Verlag, Berlin, Heidelberg (1997)

Count Corruptions, Not Users: Improved Tightness for Signatures, Encryption and Authenticated Key Exchange

Mihir Bellare[1], Doreen Riepel[2](✉), Stefano Tessaro[3], and Yizhao Zhang[1]

[1] Department of Computer Science and Engineering, University of California San Diego, La Jolla, USA
{mbellare,yiz191}@ucsd.edu

[2] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
doreen.riepel@gmail.com

[3] Paul G. Allen School of Computer Science and Engineering, University of Washington, Seattle, USA
tessaro@cs.washington.edu

Abstract. In the multi-user with corruptions (muc) setting there are $n \geq 1$ users, and the goal is to prove that, even in the face of an adversary that *adaptively* corrupts users to expose their keys, un-corrupted users retain security. This can be considered for many primitives including signatures and encryption. Proofs of muc security, while possible, generally suffer a factor n loss in tightness, which can be large. This paper gives new proofs where this factor is reduced to the number c of corruptions, which in practice is much smaller than n. We refer to this as corruption-parametrized muc (cp-muc) security. We give a general result showing it for a class of games that we call local. We apply this to get cp-muc security for signature schemes (including ones in standards and in TLS 1.3) and some forms of public-key and symmetric encryption. Then we give dedicated cp-muc security proofs for some important schemes whose underlying games are not local, including the Hashed ElGamal and Fujisaki-Okamoto KEMs and authenticated key exchange. Finally, we give negative results to show optimality of our bounds.

1 Introduction

In practice, keys can be exposed, through system infiltration by hackers or phishing attacks; a striking example is the exposure of a Microsoft signing key to the Storm-0558 threat actor in July 2023 [68]. This motivates the multi-user-with-corruptions (muc) setting, where there are $n \geq 1$ users, each holding some secret key, and the goal is to prove that, even in the face of an adversary that *adaptively* corrupts users to expose their keys, un-corrupted users retain security. (The last means their signatures remain unforgeable, ciphertexts encrypted to them retain privacy or whatever else the underlying primitive decrees.)

The good news is that proving muc-security is possible. The bad is that, in general —and in particular for canonical and standardized schemes such as the Schnorr signature scheme [66]— current reductions lose a factor of the *total* number n of users, which can be very large. This leaves implementations to either use larger security parameters (inefficient), ignore corruptions (dangerous) or turn to special schemes that, while offering tight reductions, are less efficient than the canonical schemes [2,22,36]. And meanwhile, dauntingly and disappointingly, negative results [3,48] appear to say that the factor n loss is unavoidable for the canonical schemes.

Counting corruptions. In practice, corruptions certainly happen, but we suggest that the *number* of successful corruptions will likely be small, much smaller than the total number of users. Why is this? Because key-owners, recognizing the loss they face if their keys are exposed, are taking significant steps to prevent it. Important Internet services are increasingly storing their TLS signing keys in HSMs (Hardware Security Modules), which makes them harder to expose. Breaches lead to systems being hardened to prevent further breaches. (Microsoft, for example, has taken steps to prevent a repetition of the exposure of their signing key to Storm-0558 [58].) Threshold cryptography is used to make key-exposure more difficult, a mitigation particularly popular for the signing keys securing digital wallets in cryptocurrencies. Employees and users are regularly trained to not fall prey to phishing.

Contributions in brief. Based on the above, this paper brings to muc security a new dimension, namely to view the *number of corruptions* as an adversary resource parameter. Denoting it c, we then give techniques and results that prove muc-security with tightness loss c rather than n. We refer to this as *corruption-parametrized muc (cp-muc) security*. Since c is in practice much smaller than n, cp-muc security allows theoretically-sound instantiation with practical security parameters.

We show cp-muc security for many primitives including signatures, encryption and authenticated key exchange. Our main results are *universal*, applying to *all* schemes, including the canonical ones. Existing negative results are not contradicted because they implicitly assume $n - 1$ corruptions, and we give new negative results to show the optimality of our new bounds.

Our inspiration and starting point is a technique of Coron [17] used to show security with improved bounds for the RSA-FDH signature scheme of [9]. We generalize and improve this via a modular approach. First, we define and study a primitive we call a Hamming-Weight Determined (HWD) sampler. Using HWD samplers, our general cp-muc security theorem then gives a reduction from multi-user (mu) to cp-muc security that (1) loses only a factor c and (2) holds for *any* security game satisfying a condition we call *locality*. Intuitively, this means that the game does not make use of global secrets across users, such as a global challenge bit. This directly yields cp-muc security for many schemes and goals, and avoids repeating similar proofs across them. For important games that are not local, we go on to give dedicated proofs of cp-muc security. In particular, we do so (for security games with a global challenge bit) for the Hashed ElGamal and Fujisaki-Okamoto KEMs.

Broader context. The idea of assuming a bound c on the number of corruptions, amongst some larger number n of users, arises, and is indeed the basis for security, in some other and prior contexts in cryptography. The most prominent example is certainly secret sharing [13,67], where c is the threshold. Thence it enters multi-party computation [34] and threshold cryptography [20]. With cp-muc, we are bringing this classical perspective to a broader setting, and to basic primitives like signatures, encryption and authenticated key exchange.

1.1 Spotlight on Signatures

Our techniques and results are general and yield improvements for many goals and schemes. The ideas are however best introduced via a concrete example. We'll use signatures, for a number of reasons. First, there are, in the wild, over 250 million TLS signing keys [24]. Exposure is a real threat, making as-tight-as-possible muc-security of immediate practical interest. Second, muc-secure signatures with as-tight-as-possible security are an assumption and current bottleneck for tightly-secure authenticated key exchange [16,19,23,36]. Our proofs of cp-muc security for signature schemes will help to fill these gaps.

The settings. Recall that classical definitions of security consider a single-user (su) setting. For a signature scheme Sig, recall this means that the game (called UF, for unforgeability) generates a single pair (vk, sk) consisting of a verification key and associated signing key. The adversary gets an oracle Sign to sign a message of its choice under sk, and wins if it produces a new message M and a valid signature of M under vk [35]. In the multi-user (mu) setting [4], there are $n \geq 1$ users. For signatures [57], the game generates n independent key pairs $(vk_1, sk_1), \ldots, (vk_n, sk_n)$. The signing oracle Sign now takes a user index i in addition to a message and returns a signature of the message under sk_i, and to win the adversary would point to some user i and produce a new message M, and a valid signature for it, under vk_i. The multi-user with corruptions (muc) game augments the mu game with an oracle Corrupt that takes a user index i and returns its secret key sk_i. The signing oracle Sign is as before, and the winning forgery is required to be for an uncorrupted user.

By $\epsilon_{\mathsf{Sig}}^{\text{uf-su}}$, $\epsilon_{\mathsf{Sig}}^{\text{uf-mu-}n}$ and $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}n}$ we denote the adversary advantage (success probability) in the su, mu and muc games, respectively. Of course, this quantity depends on the running time, but we will consider reductions that preserve this time and omit it from the notation here. The body of the paper is more precise.

Universal results. We are interested, first, in results that are *universal*, meaning hold for *all* signature schemes Sig. Indeed, universal results are valuable both theoretically (in terms of understanding the notions) and in practice (they apply to all schemes, including already implemented and standardized ones). An archetypal such result [4,57] (shown by a hybrid, also called guess-then-simulate, argument) is that $\epsilon_{\mathsf{Sig}}^{\text{uf-mu-}n} \leq n \cdot \epsilon_{\mathsf{Sig}}^{\text{uf-su}}$. The proof turns out to be able to easily handle corruptions, yielding the only known universal result for muc security:

$$\underline{\text{P1}} : \ \forall \mathsf{Sig} \ : \ \ \epsilon_{\mathsf{Sig}}^{\text{uf-muc-}n} \leq n \cdot \epsilon_{\mathsf{Sig}}^{\text{uf-su}} \ . \tag{1}$$

We ask if alternative universal results are possible. Given that mu security has been extensively studied [11,33,41,45,50,53,62], we suggest to use it, rather than su, as the starting point. Let e be Euler's number. Our first universal result is then:

$$\underline{\text{N1}}: \ \forall \mathsf{Sig} : \ \epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)} \leq e \cdot (c+1) \cdot \epsilon_{\mathsf{Sig}}^{\text{uf-mu-}n} \ . \tag{2}$$

The term on the left above is the cp-muc advantage where the adversary is restricted to c queries to oracle CORRUPT. Equation (2) says that, in moving from the setting with no corruptions to the setting with c corruptions, adversary advantage grows by at most a factor about c, *regardless of the number n of users* and *for all signature schemes* Sig. But we can do better. In Eq. (2), muc security for n users (and c corruptions) is provided assuming mu security for the same number n of users. It turns out, curiously, that we can (substantially) reduce the number of users, now denoted m, for which mu security is assumed. Our second universal result (Theorem 6) is:

$$\underline{\text{N2}}: \ \forall \mathsf{Sig} : \ \epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)} \leq e \cdot (c+1) \cdot \epsilon_{\mathsf{Sig}}^{\text{uf-mu-}m} \ \text{ for } m = \lfloor (n-1)/(c+1) \rfloor . \tag{3}$$

How and when the new N1, N2 results yield improvements over the prior P1 may not be obvious due to the starting points being different (mu for the former and su for the latter). The following will clarify this.

<u>SPECIALIZED RESULTS.</u> These are results that hold for *particular* (but not all) schemes. Let's call Sig *tightly-mu-secure* if (roughly) $\epsilon_{\mathsf{Sig}}^{\text{uf-mu-}n} = \epsilon_{\mathsf{Sig}}^{\text{uf-su}}$. In an important class of specialized results, prior work [11,33,41,45,50,53,62] shows that many schemes —including the Schnorr scheme [11,50]— have tight mu security. The proofs, however, exploit algebraic self-reducibility amongst keys and cannot tolerate corruptions, so that, even for these schemes, for muc security, Eq. (1) remains the best known bound. We offer a substantial improvement; Eq. (2) implies that

$$\underline{\text{N3}}: \ \text{For all tightly-mu-secure } \mathsf{Sig}: \ \epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)} \leq e \cdot (c+1) \cdot \epsilon_{\mathsf{Sig}}^{\text{uf-su}} \ . \tag{4}$$

That is, the muc advantage degrades by at most a factor about c even relative to the (standard) su advantage, again regardless of n. Equation (4) holds in particular for the Schnorr signature scheme.

While it would be desirable to show that su security implies cp-muc security with a loss c in general and for all schemes, such a statement does not seem possible without additional assumption about the scheme. Therefore, our results can be viewed as a middle ground, either (using N3) applying to tightly-mu-secure schemes or (using N2) reducing to mu security for a number of users m substantially smaller than n.

<u>NUMERICAL EXAMPLE.</u> Let Sig be the Schnorr scheme over a size p elliptic curve group $\mathbb{G}$. Suppose we target 128-bit security for $n = 2^{30}$ users with $c = 2^{10}$

corruptions, and let t be the running-time of the adversary. Assuming discrete-logarithm computation over $\mathbb{G}$ is the best attack, we have $\epsilon_{\mathsf{Sig}}^{\text{uf-su}} \leq t^2/p$, which by the prior P1 result of Eq. (1) yields $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}n} \leq nt^2/p$. Now 128-bit security requires $nt^2/p \leq t/2^{128}$ or $p = 2^{128} \cdot nt$. This means we use a group $\mathbb{G}_1$ of size $p_1 = 2^{128} \cdot nt$. Meanwhile the tight mu security of Sig implies $\epsilon_{\mathsf{Sig}}^{\text{uf-mu-}n} \leq t^2/p$ which by our new N1 result of Eq. (2) (and ignoring the e factor) yields $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)} \leq ct^2/p$, and $ct^2/p \leq t/2^{128}$ now yields $p = 2^{128} \cdot ct$, so we use a group $\mathbb{G}_2$ of size $p_2 = 2^{128} \cdot ct$. Now note that $p_2/p_1 = c/n = 2^{-20}$ so we have dropped the group size by a factor of 2^{20} while retaining security. For example if $p_1 = 2^{256}$ then $p_2 = 2^{236}$. Exponentiation takes time cubic in the logarithm of the group size, so in this case we conclude that the cost of signing or verifying for Sig over $\mathbb{G}_1$ is $(236/256)^3 = 0.78$ times that in $\mathbb{G}_2$, meaning we have reduced the running time of the implemented scheme by 22%, a significant gain in practice, while retaining security.

STANDARDS AND KEY EXCHANGE. The best muc-security bound we have for the standardized RSA-SSAPSS [61], EdDSA [12,49,61] and ECDSA [61] signature schemes is the generic one of Eq. (1) with its factor n loss. This lead tight proofs for the TLS 1.3 key exchange [19,23] to simply assume tight muc security — meaning $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}n} = \epsilon_{\mathsf{Sig}}^{\text{uf-su}}$— for these schemes. Can we do better? Unfortunately, Result N3 (Eq. (4)) will not help since these schemes have evaded proofs of tight mu security. However, Result N2 (Eq. (3)) is helpful here. It says that assuming $\epsilon_{\mathsf{Sig}}^{\text{uf-mu-}m} = \epsilon_{\mathsf{Sig}}^{\text{uf-su}}$ for some small number m of users, we get $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)} \leq c \cdot \epsilon_{\mathsf{Sig}}^{\text{uf-su}}$. While this is still a non-standard assumption, it is better than directly assuming $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}n} = \epsilon_{\mathsf{Sig}}^{\text{uf-su}}$. We only need to assume mu (rather than muc) security, and that too for a small number of users m.

TIGHT MUC SECURITY. The absence of results better than P1 (Eq. (1)) for standardized schemes lead researchers to develop new signature schemes that are tightly muc secure [2,22,32,36,38,63]. (As above this means $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}n} = \epsilon_{\mathsf{Sig}}^{\text{uf-su}}$.) In terms of just the bound, this is better than cp-muc security. But canonical schemes (Schnorr and standardized ones) are not known to be in this class so there is no improvement for in-use signatures or TLS 1.3 key exchange. Moreover, the key sizes and computation time of the new schemes is larger than that of the canonical schemes. Hence, cp-muc security is a pragmatic alternative.

TECHNIQUES. We outline the simplest case of our proof technique, specialized to signatures and for the weaker of our two universal results, namely N1 (Eq. (2)). Given a cp-muc adversary $\mathcal{A}$ with advantage $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)}$, we want to construct a mu adversary $\mathcal{B}$, with advantage $\epsilon_{\mathsf{Sig}}^{\text{uf-mu-}n}$ and about the same running time as $\mathcal{A}$, such that Eq. (2) holds. Letting p be a parameter in the range $0 \leq p \leq 1$, adversary $\mathcal{B}$ picks a vector $(c_1, \ldots, c_n)$ of independent, p-biased bits. (That is, each c_i is 1 with probability p and 0 with probability $1-p$.) We'll say user i is red if $c_i = 1$ and let $R \subseteq \{1, \ldots, n\}$ be the set of red users; correspondingly i is blue if $c_i = 0$ and B is the set of blue users. Now, $\mathcal{B}$ runs $\mathcal{A}$. In answering $\mathcal{A}$'s oracle queries, $\mathcal{B}$ simulates red users directly and forwards queries for blue users to its own mu game. In more detail, $\mathcal{B}$ generates a signing-verifying key pair (vk_i, sk_i) for each $i \in R$, allowing it to easily answer both SIGN and CORRUPT queries to i. It answers SIGN queries for $i \in B$ via its own SIGN oracle. The difficulty is a CORRUPT(i) query for $i \in B$; adversary $\mathcal{B}$ has no way to answer this, and aborts. If it does not abort, then $\mathcal{A}$ outputs a user j (called the forgery victim) and a forgery under vk_j. If $j \in B$ then $\mathcal{B}$ returns this to win its mu game, else it aborts. Let GD be the event that $i \in R$ for all CORRUPT(i) queries of $\mathcal{A}$ and also the forgery victim j is in B. The probability of GD can be computed as $f(p) = p^c(1-p)$ (Theorem 2) and a careful analysis (that we make precise via Lemma 2) shows that GD is independent of the success of $\mathcal{A}$, so that $\epsilon_{\mathsf{Sig}}^{\text{uf-mu-}n} \geq f(p) \cdot \epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)}$. Calculus shows that $f(p)$ is maximized at $p = 1 - 1/(c+1)$ where $f(p) \geq 1/(e(c+1))$ (Theorem 2), yielding Eq. (2). For some intuition, note that with this choice the expected number of red users is $(cn-1)/(c+1)$, meaning a high number of red users are needed to successfully answer the small number c of CORRUPT queries.

This adapts Coron's [17] proof of the security of the RSA-FDH signature scheme. However, his setting has only one user and no secret-key exposing corruptions; what for us are "users" and "corruptions" are for Coron messages queried to the random oracle and signing queries, respectively.

THE PATH AHEAD. We generalize and improve this with a modular approach. First, we introduce and study HWD samplers, as a way to find the best way to sample the vector $(c_1, \ldots, c_n)$. Second, we give a framework and general cp-muc security theorem that shows how to use any HWD sampler to promote mu security to cp-muc security for a large class of games satisfying a condition we call locality. Numerous applications, as well as improvements such as Eq. (3), are then obtained, some directly, some with more dedicated work.

1.2 HWD Samplers and General cp-muc Theorem

HWD SAMPLERS. We ask whether the above manner of choosing $(c_1, \ldots, c_n)$ is optimal. The answer is no; we can do better. To get there, we start with an abstraction, viewing the vector $(c_1, \ldots, c_n) \leftarrow\$ \mathsf{D}(n, c)$ as being chosen by an algorithm D that we call a sampler. We identify, as a sufficient condition on D for the (above) simulation to work, that the probability of a vector $(c_1, \ldots, c_n)$ depends only on its Hamming weight, and call such samplers Hamming-Weight Determined (HWD). Section 3 associates to any HWD sampler a success probability α and an error probability β. Continuing to use signatures as an example, Theorem 4 implies that $\epsilon_{\mathsf{Sig}}^{\text{uf-muc-}(n,c)} \leq (1/\alpha) \cdot \epsilon_{\mathsf{Sig}}^{\text{uf-mu-}m}$ for an m that grows with β, so we want to make α large and β small. Now the way of sampling $(c_1, \ldots, c_n)$ discussed in Sect. 1.1 corresponds to a particular choice of D, that we call the biased-coin sampler and analyze in Theorem 2; it has good success probability but unfortunately high error probability. We then give a new sampler that we call the fixed-set-size sampler. Theorem 3 shows that it has not only optimal success probability but zero error probability, which is crucial to obtaining the improved Eq. (3). Figure 3 shows some numbers.

General framework and theorem. We now ask how far these techniques will generalize beyond signatures. We seek to show a result of the form "For all security games in a certain class, mu security can be promoted to cp-muc security with loss only a factor about c." To do this rigorously, we first (in Sect. 4) define something we call a *formal security specification* (FSS). Denoted $\mathsf{\Pi}$, it is simply an algorithm that, given a scheme Sch, specifies the initializations and oracle-response computations (including for Corrupt queries) for the intended target notion of security for Sch. This leads naturally to an actual (code-based) game $\mathbf{G}^{\mathsf{\Pi}}_{\mathsf{Sch}}$ defining su security (Fig. 4), and thence to games $\mathbf{G}^{\mathsf{\Pi}\text{-mu-}n}_{\mathsf{Sch}}$ and $\mathbf{G}^{\mathsf{\Pi}\text{-muc-}(n,c)}_{\mathsf{Sch}}$ capturing mu and cp-muc security, respectively, where n is the number of users and c the number of allowed corruptions (Fig. 5). For example, Fig. 4 shows an FSS UF such that, when $\mathsf{Sch} = \mathsf{Sig}$ is a signature scheme, the corresponding games are exactly the standard su, mu and cp-muc games for signatures discussed above. The su, mu and muc versions of many other notions in the literature can in this way be recovered by simply giving a single FSS for the notion, as we will see. Let $\epsilon^{\mathsf{\Pi}\text{-su}}_{\mathsf{Sch}}$, $\epsilon^{\mathsf{\Pi}\text{-mu-}n}_{\mathsf{Sch}}$ and $\epsilon^{\mathsf{\Pi}\text{-muc-}(n,c)}_{\mathsf{Sch}}$ denote the su, mu and muc adversary advantages, respectively, for FSS $\mathsf{\Pi}$ with scheme Sch.

In Sect. 4 we define a condition on an FSS that we call *locality*; roughly it means that in the mu and muc settings, different users share no common secret unknown to the adversary. This is what is needed for our technique to work. The main result is Theorem 4, showing how to use any HWD sampler D to promote mu to cp-muc for any local FSS $\mathsf{\Pi}$ for a scheme Sch, and giving bounds in terms of quantities related to D. The proof uses game playing including a crucial use of the Second Fundamental Lemma (Lemma 2). For applications however, it is easier to use Theorem 5, which sets D to the optimal fixed-set-size sampler to say that for any local FSS $\mathsf{\Pi}$ for a scheme Sch, we have

$$\epsilon^{\mathsf{\Pi}\text{-muc-}(n,c)}_{\mathsf{Sch}} \leq e \cdot (c+1) \cdot \epsilon^{\mathsf{\Pi}\text{-mu-}m}_{\mathsf{Sch}} \quad \text{for } m = \lfloor (n-1)/(c+1) \rfloor. \tag{5}$$

As an example, FSS UF for signature schemes is local, so this immediately yields Eqs. Equations (2), (3). We now turn to further applications.

1.3 Applications

An overview of our applications is in Fig. 1. The results are of the form $\epsilon^{\text{TG-}(n,c)}_{\mathsf{TS}} \leq c \cdot \epsilon^{\text{SG-}m}_{\mathsf{SS}}$ where $\text{TG}, \text{SG}, \mathsf{TS}, \mathsf{SS}$ are the target goal, starting goal, target scheme and starting scheme, respectively. That is, TG-security of TS for n users and c corruptions is shown with loss c assuming SG-security for SS for m users. What are "users" and "corruptions" depends on the goal, showcasing the breadth of our framework. The last column indicates the application type. Direct (D) means we show that an underlying FSS is local and apply Theorem 5. Indirect (I) means the FSS is *not* local but we can nevertheless show the result via a dedicated proof that uses the general theorem in an intermediate step.

Direct applications. **D1** is the result for signatures (Theorem 6). **D2** is for IND-CCA security of KEMs in the multi-challenge-bit (mb) setting [43] (Theorem 7). **D3** is for OW-PCVA (one-way security under plaintext checking and

	Target		**Start**		m	Type
	TG	TS	SG	SS		
D1	UF-muc	Sig	UF-mu	Sig	$\lfloor (n-1)/(c+1) \rfloor$	D
D2	CCA-MB-muc	KEM	CCA-MB-mu	KEM	$\lfloor (n-1)/(c+1) \rfloor$	D
D3	OW-PCVA-muc	PKE	OW-PCVA-mu	PKE	$\lfloor (n-1)/(c+1) \rfloor$	D
D4	AE-MB-muc	SE	AE-MB-mu	SE	$\lfloor (n-1)/(c+1) \rfloor$	D
SB1	CCA-SB-muc	HEG	St-CDH	$(\mathbb{G}, p, g)$	1	I
SB2	CCA-SB-muc	KEM	CPA-SB-mu	PKE	$\lfloor (n-1)/(c+1) \rfloor$	I
KE1	IND-WFS	CCGJJ	St-CDH	$(\mathbb{G}, p, g)$	$\lfloor (n-1)/(c+1) \rfloor$	I
KE2	IND-FFS	AKE	UF-mu	Sig	$\lfloor (n-1)/(c+1) \rfloor$	I
SO1	SIM-SO-CCA	DH-PKE	St-CDH	$(\mathbb{G}, p, g)$	1	I
SO2	SIM-SO-CCA	RSA-PKE	RSA	RSAGen	1	I

Fig. 1. Overview of our applications: We show that $\epsilon_{\mathsf{TS}}^{\mathrm{TG}\text{-}(n,c)} \leq c \cdot \epsilon_{\mathsf{SS}}^{\mathrm{SG}\text{-}m}$ where TS, TG are the target scheme and goal, and SS, SG are the starting scheme and goal. **D1-D4** are direct applications of our general cp-muc security theorem; the rest are indirect. **SB1-SB2** relate to encryption in the single-bit setting. **KE1-KE2** are for (authenticated) key exchange. **SO1-SO2** are for selective opening security.

ciphertext validity attacks) security of PKE [44] (Theorem 8), which will be used in our results for the FO transform. **D4** is for (nonce-based) authenticated encryption in the mb setting (Theorem 9), demonstrating the generality of our approach in that our results also apply in the symmetric setting. In each case we observe that the underlying FSSs, denoted UF, CCA-MB, OW-PCVA and AE-MB respectively, are local, yielding the results via Theorem 5. We stress that these rows are purely illustrative; there are far more such applications than we can exhaustively list.

Indirect applications. In the definition of mu security for encryption (KEM, PKE) from [4], there is a single challenge bit across all users. Therefore, the underlying FSS CCA-SB is *not* local. However, this is recognized as the leading and stronger definition [31] so rather than give up on it, we ask what one can show. Our answer is to show cp-muc security for particular, important schemes via dedicated proofs, making use of our general theorem for intermediate steps. We first prove such a result (**SB1**, Theorem 10) for the hashed ElGamal KEM HEG, assuming strong computational Diffie-Hellman (St-CDH) [1]. Then, we use the modular FO transform [44] to get a more general result (**SB2**, Theorem 11). In particular, we show how to turn a CPA-SB-mu PKE scheme into a CCA-SB-muc KEM. For this, we first go tightly from CPA-SB-mu to OW-PCVA-mu. Next, we exploit locality of the corresponding FSS OW-PCVA and use Theorem 5 to go to OW-PCVA-muc. Finally, we go tightly from the latter to CCA-SB-muc.

As a natural extension, we then turn to authenticated key exchange (AKE). Security games for AKE (e. g., [8]) also use a single challenge bit and are not local, but we can still give several cp-muc security results. The Diffie-Hellman based CCGJJ protocol [16] is shown in [16] to achieve weak forward secrecy (wfs) [52], with a factor n loss from $\mathsf{St\text{-}CDH}$. We show (**KE1**) that by considering our corruption-parameterized approach for AKE, we can reduce the loss to the number c of corruptions. Previous work on tightly-secure AKE [32,36,56], including analyses of the TLS protocol [19,23], all use muc-secure signatures as a building block. Our results on signatures allow AKE security based on UF-mu secure schemes (**KE2**) which yields tightness improvements when using the canonical and standardized signature schemes actually used in practice.

We then extend our indirect results to applications in a quite different setting, looking at selective opening security for PKE. In simulation-based selective opening ($\mathsf{SIM\text{-}SO\text{-}CCA}$) security for PKE [5,14], the adversary can reveal randomness underlying encrypted messages. Practical Diffie-Hellman and RSA-based PKE schemes are shown in [42] to be $\mathsf{SIM\text{-}SO\text{-}CCA}$, with loss factor the total number n of ciphertexts encrypted. Modeling randomness reveal as a "corruption" and each encryption as a "user" in our framework allows us (with some extra steps) to reduce the loss to the number c of openings. Finally, of pedagogic interest but no novelty, we show how our framework can be used to recover Coron's result [17].

1.4 Optimality Results

We are interested in the optimality of our results, namely to show that the factor c loss is optimal for a number of schemes and games. Previous work studied optimality based on *non-interactive* complexity assumptions [3,37], *bounded-round* assumptions [59] or tailored to specific primitives or protocols [16,30,48]. However, a mu security game is mostly interactive and not bounded. Similar to our positive result, we aim for a general result capturing a variety of games.

Witness recovery games. More concretely, we consider the goal of proving security, via black-box reductions, for a *recovery* game $\mathbf{G}^{\mathsf{REC}\text{-muc-}(n,c)}_{\mathsf{Rel}}$ parameterized by an efficiently verifiable relation Rel. After learning n inputs $x_1, \ldots, x_n$, the adversary is allowed to learn witnesses (with respect to Rel) for up to c of them, and finally wins if it recovers a witness for one of the remaining inputs. As already observed by [3], several mu security games $\mathbf{G}'$ allow to define a suitable relation Rel such that $\mathbf{G}^{\mathsf{REC}\text{-muc-}(n,c)}_{\mathsf{Rel}}$ can be seen as a *restriction* of the set of allowable adversaries in $\mathbf{G}'$. Examples include relations for PKE and signatures, where the decryption resp. signing key serves as the witness.

Stateless games. Our first general result shows that any black-box reduction from an interactive *stateless* game $\mathbf{G}$ to $\mathbf{G}^{\mathsf{REC}\text{-muc-}(n,c)}_{\mathsf{Rel}}$ must incur a loss $c + 1$ when Rel is efficiently randomizable (Theorem 12). The same loss is hence also necessary if we instead reduce to a game $\mathbf{G}'$ for which $\mathbf{G}^{\mathsf{REC}\text{-muc-}(n,c)}_{\mathsf{Rel}}$ can be seen as an adversarial restriction. Here, by stateless we mean that the game initially sets a state, but then oracle queries do not alter it. This abstraction serves a trade-off between simplicity and generality. Indeed, several games are stateless, such as CPA security for public-key encryption in both the multi-challenge-bit as well as in the single-challenge-bit setting.

However, since many games of interest are not stateless, we extend our result to relax the stateless condition on $\mathbf{G}$. This generalization allows the analysis of stateful games and, more concretely, we show optimality of the loss factor c for the IND-CCA game for PKE and the strong unforgeability game for randomized signatures. (Assuming ciphertexts and signatures have sufficient entropy.)

2 Preliminaries

<u>Notation.</u> If $\boldsymbol{w}$ is a vector then $|\boldsymbol{w}|$ is its length (the number of its coordinates) and $\boldsymbol{w}[i]$ is its i-th coordinate, where we start with $i = 1$. Strings are identified with vectors over $\{0,1\}^*$, so that $|Z|$ denotes the length of a string Z and $Z[i]$ denotes its i-th bit. By ε we denote the empty string or vector. By $x \| y$ we denote the concatenation of strings x, y. If x, y are equal-length strings then $x \oplus y$ denotes their bitwise xor. If S is a finite set, then $|S|$ denotes its size. For integers $a \leq b$ we let $[a..b]$ be shorthand for $\{a, \ldots, b\}$. The notation $[\![B]\!]$ for a boolean statement B returns true if B is true and false otherwise. When we write e, we mean Euler's number.

If X is a finite set, we let $x \leftarrow\!\!\$\; X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. If A is an algorithm, we let $y \leftarrow\!\!\$\; A[\mathrm{O}_1, \ldots](x_1, \ldots)$ denote running A on inputs $x_1, \ldots$, with oracle access to $\mathrm{O}_1, \ldots$, and assigning the output to y. We let $\mathbf{Out}(A[\mathrm{O}_1, \ldots](x_1, \ldots))$ denote the set of all possible outputs of A on this run. We use $\perp$ (bot) as a special symbol to denote rejection, and it is assumed to not be in $\{0,1\}^*$. We may consider an algorithm A that takes some input in and a current state, and returns an output out and updated state, written $(out, st) \leftarrow\!\!\$\; A(in, st)$. We may refer to such an algorithm as stateful, but note that A is, syntactically, just an algorithm like any other. There is no implicit state maintained by the algorithm; it is the responsibility of the game executing A to maintain the state variable st, which it will do explicitly.

<u>Games.</u> We use the code-based game-playing framework of BR [10]. By $\Pr[\mathrm{G}(\mathcal{A}) \Rightarrow y]$ we denote the probability that the execution of game G with adversary $\mathcal{A}$ results in the game output (what is returned by Fin) being y, and write just $\Pr[\mathrm{G}(\mathcal{A})]$ for $\Pr[\mathrm{G}(\mathcal{A}) \Rightarrow \mathsf{true}]$. Different games may have procedures (oracles) with the same names. If we need to disambiguate, we may write G.O to refer to oracle O of game G. In games, integer variables, set variables, boolean variables and string variables are assumed initialized, respectively, to 0, the empty set $\emptyset$, the boolean false and $\perp$. For the following, recall that games G, H are identical-until-bad if their code differs only in statements that follow the setting of flag bad to true [10].

Lemma 1. [First Fundamental Lemma of Game Playing [10]] *Let* G, H *be identical-until-*bad *games. Then for any adversary* $\mathcal{A}$ *we have*

$$|\Pr[\mathrm{G}(\mathcal{A})] - \Pr[\mathrm{H}(\mathcal{A})]| \leq \Pr[\mathrm{H}(\mathcal{A}) \textit{ sets } \mathsf{bad}] = \Pr[\mathrm{G}(\mathcal{A}) \textit{ sets } \mathsf{bad}] .$$

Lemma 2. [Second Fundamental Lemma of Game Playing [6]] *Let* G, H *be identical-until-*bad *games and let* GD *be the event that* bad *is not set to* true *in the execution of these games with adversary* $\mathcal{A}$. *Then*

$$\Pr[\mathrm{G}(\mathcal{A}) \wedge \mathrm{GD}] = \Pr[\mathrm{H}(\mathcal{A}) \wedge \mathrm{GD}] \,.$$

<u>Signature schemes.</u> A signature scheme Sig allows generation of a verifying key vk and corresponding signing key sk via $(vk, sk) \leftarrow\$ \mathsf{Sig.Kg}$. A signature of a message M is produced as $\sigma \leftarrow\$ \mathsf{Sig.Sign}(sk, M)$. Verification returns a boolean $d \leftarrow \mathsf{Sig.Vf}(vk, M, \sigma)$. Correctness asks that for all M we have $\mathsf{Sig.Vf}(vk, M, \mathsf{Sig.Sign}(sk, M)) = \mathsf{true}$ with probability 1, where the probability is over the choice of keys and the coins of the signing algorithm, if any. To measure security we define the uf advantage of an adversary $\mathcal{A}$ as $\mathbf{Adv}^{\mathsf{UF\text{-}su}}_{\mathsf{Sig}}(\mathcal{A}) = \Pr[\mathbf{G}^{\mathsf{UF}}_{\mathsf{Sig}}(\mathcal{A})]$ where the game is shown on the bottom right of Fig. 4.

<u>Public-key encryption schemes.</u> A public-key encryption (PKE) scheme PKE allows generation of a public encryption key ek and corresponding secret decryption key dk via $(ek, dk) \leftarrow\$ \mathsf{PKE.Kg}$. We denote the message space by $\mathsf{PKE.MS}$ and require that it is a length-closed set (i. e., for any $m \in \mathsf{PKE.MS}$ it is the case that $\{0,1\}^{|m|} \subseteq \mathsf{PKE.MS}$). Encryption of a message $M \in \mathsf{PKE.MS}$ is via $C \leftarrow\$ \mathsf{PKE.Enc}(ek, M)$. Decryption $M \leftarrow \mathsf{PKE.Dec}(ek, dk, C)$ returns a value $M \in \{0,1\}^* \cup \{\bot\}$. Correctness asks that $\Pr[\mathsf{PKE.Dec}(dk, \mathsf{PKE.Enc}(ek, M)) = M] = 1$ for every $(ek, dk) \in \mathbf{Out}(\mathsf{PKE.Kg})$ and every $M \in \mathsf{PKE.MS}$. We will consider several security definitions; they will be given in the full version [7].

<u>Key-encapsulation mechanisms.</u> A key-encapsulation mechanism (KEM) KEM allows generation of a public encapsulation key ek and corresponding secret decapsulation key dk via $(ek, dk) \leftarrow\$ \mathsf{KEM.Kg}$. There is no message. Instead, encapsulation $\mathsf{KEM.Encaps}(ek)$ returns a symmetric key $K \in \{0,1\}^{\mathsf{KEM.kl}}$ and a ciphertext C encapsulating it. Decapsulation $\mathsf{KEM.Decaps}(dk, C)$ returns a value $K \in \{0,1\}^{\mathsf{KEM.kl}} \cup \{\bot\}$. Correctness asks that $\Pr[\mathsf{KEM.Decaps}(dk, C) = K] = 1$ for every $(ek, dk) \in \mathbf{Out}(\mathsf{KEM.Kg})$, where the probability is over $(C, K) \leftarrow\$ \mathsf{KEM.Encaps}(ek)$. Security definitions are in Sect. 5.

3 HWD Samplers

We introduce and define HWD samplers and their success and error probabilities. We give and analyze a natural HWD sampler called the biased-coin sampler and then give an optimal HWD sampler called the fixed-set-size sampler.

<u>HWD samplers.</u> A *sampler* is an algorithm D that on input a positive integer n and a non-negative integer $c < n$ returns an n-bit string $s \leftarrow\$ \mathsf{D}(n, c)$. In our application, n will be the number of users and c will be the number of corruptions. For $s \in \{0,1\}^n$, let $R(s) = \{\, i \in [1..n] : s[i] = 1 \,\}$ and $B(s) = \{\, i \in [1..n] : s[i] = 0 \,\}$. We think of sampling $s \leftarrow\$ \mathsf{D}(n, c)$ as coloring the points $1, 2, \ldots, n$, with point $i \in [1..n]$ colored red if $s[i] = 1$ and colored blue if $s[i] = 0$, so that $R(s)$ and $B(s)$ are the sets of red and blue points, respectively.

We say that sampler D is *Hamming-weight determined* (HWD) if for all n, c and all $s_1, s_2 \in \{0,1\}^n$ we have: if $\mathbf{w}_{\mathrm{H}}(s_1) = \mathbf{w}_{\mathrm{H}}(s_2)$ then s_1 and s_2 have the same probability of arising as outputs of $\mathsf{D}(n, c)$. This will allow our applications. Note that this condition is true if and only if there is a function W such that $\Pr[s = s' : s' \leftarrow\!\!\$\ \mathsf{D}(n, c)] = \mathrm{W}(n, c, \mathbf{w}_{\mathrm{H}}(s))$ for all $s \in \{0,1\}^n$ and all n, c. We refer to W as D's *Hamming-weight probability function*.

As another way of understanding this, if $\pi : [1..n] \to [1..n]$ is a permutation, let $\pi(s) \in \{0,1\}^n$ denote the string whose j-th bit is $s[\pi(j)]$ for $j \in [1..n]$. That is, the bits of s are permuted according to π. Now say that D is *permutation invariant* if for all n, c, all $s \in \{0,1\}^n$ and all permutations $\pi : [1..n] \to [1..n]$ we have that s and $\pi(s)$ have the same probability of arising as outputs of $\mathsf{D}(n, c)$. Then it is easy to see that D is permutation invariant if and only if it is Hamming-weight determined.

For a set $C \subseteq [1..n]$ of size $c < n$, and $i \in [1..n] \setminus C$, we let

$$\mathbf{P}_{\mathsf{D},n,c}(C, i) = \Pr[\, i \in B(s) \text{ and } C \subseteq R(s) \ : \ s \leftarrow\!\!\$\ \mathsf{D}(n, c)] \,. \tag{6}$$

This is the probability that $s \leftarrow\!\!\$\ \mathsf{D}(n, c)$ colors the points in C red and colors i blue. (How points not in $C \cup \{i\}$ are colored does not matter.) Our applications need a lower bound on it. Towards this, we define the *sampler success probability*

$$\mathbf{P}^*_{\mathsf{D}}(n, c) = \textstyle\sum_{j=0}^{n-c-1} \mathrm{W}(n, c, c + j) \cdot \binom{n-c-1}{j} \,. \tag{7}$$

The following says $\mathbf{P}_{\mathsf{D},n,c}(C, i)$ is determined by the success probability and thus in particular independent of the particular choices of C, i. The proof is in the full version [7].

Theorem 1. *Let D be a Hamming-weight determined sampler with Hamming-weight probability function W. Suppose $0 \le c < n$. Then for any size c set $C \subseteq [1..n]$ and any $i \in [1..n] \setminus C$ we have $\mathbf{P}_{\mathsf{D},n,c}(C, i) = \mathbf{P}^*_{\mathsf{D}}(n, c)$.*

Theorem 1 will be used in the proof of Theorem 4, our general cp-muc security result. Here it allows us to focus on the success probability as defined by Eq. (7). Now define the *error probability* of sampler D via

$$\mathbf{R}_{\mathsf{D}}(n, c, m) = \Pr[\, |B(s)| > m \ : \ s \leftarrow\!\!\$\ \mathsf{D}(n, c)] \,. \tag{8}$$

We want to upper bound this, which in our application will allow m to be the number of users for which mu security without corruptions is assumed.

<u>Biased-coin sampler.</u> We extract from the Coron technique [17] a sampler $\mathsf{D\text{-}BC}^x$ that we call the biased-coin sampler and show on the left in Fig. 2. Here $x \in [0, 1]$ is a probability and $b \leftarrow_x \{0,1\}$ returns a x-biased coin $b \in \{0,1\}$, meaning $\Pr[b = 1] = x$ and $\Pr[b = 0] = 1 - x$. We let $\mathsf{D\text{-}BC} = \mathsf{D\text{-}BC}^x$ for $x = 1 - 1/(c+1)$ and refer to this as the optimal biased-coin sampler. Theorem 2 below justifies this name by showing that $\mathsf{D\text{-}BC}$ maximizes the success probability across samplers in the biased-sampler class. It also gives a good lower bound on this success probability and bounds the sampler error probability.

```
Sampler $\mathsf{D\text{-}BC}^x$:
For $j = 1, \ldots, n$ do
    $s[j] \leftarrow_x \{0,1\}$
Return $s$
```

```
Sampler $\mathsf{D\text{-}FX}^t$:
$T \leftarrow\!\!\$\ \mathcal{P}(n,t)$
For $j = 1, \ldots, n$ do
    If $j \in T$ then $s[j] \leftarrow 1$ else $s[j] \leftarrow 0$
Return $s$
```

Fig. 2. Our Hamming-Weight Determined Samplers. **Left:** Biased-coin sampler with probability parameter $x \in [0,1]$. **Right:** Fixed set size sampler with set-size parameter $t \in [0..n-1]$.

Theorem 2. *Let $n, m \geq 1$ and $c \geq 0$ be integers such that $n/(c+1) < m$ and $c < n$. Let $\mathsf{D\text{-}BC}^x$ be the biased-coin sampler associated to $x \in [0,1]$ as per Fig. 2. Then $\mathsf{D\text{-}BC}^x$ is Hamming-weight determined and $\mathbf{P}^*_{\mathsf{D\text{-}BC}^x}(n,c) = x^c(1-x)$. Furthermore $\mathbf{P}^*_{\mathsf{D\text{-}BC}}(n,c) \geq \mathbf{P}^*_{\mathsf{D\text{-}BC}^x}(n,c)$ for all $x \in [0,1]$ and*

$$\mathbf{P}^*_{\mathsf{D\text{-}BC}}(n,c) = \left(1 - \frac{1}{c+1}\right)^c \frac{1}{c+1} \geq \frac{1}{e} \cdot \frac{1}{c+1} \, . \tag{9}$$

Letting $a = m - n/(c+1)$ we also have $\mathbf{R}_{\mathsf{D\text{-}BC}}(n,c,m) \leq e^{-a^2/2n}$ if $m < n$ and $\mathbf{R}_{\mathsf{D\text{-}BC}}(n,c,m) = 0$ if $m \geq n$.

The proof is in the full version [7]. Here we sketch the main ideas. We observe that $\mathsf{D\text{-}BC}^x$ is HWD with Hamming-weight probability function $\mathrm{W}(n,c,\ell) = x^\ell(1-x)^{n-\ell}$. Let $f(x) = x^c(1-x)$ and let $p = c/(c+1) = 1 - 1/(c+1)$. Then Eq. (7) is used to show that $\mathbf{P}^*_{\mathsf{D\text{-}BC}^x}(n,c) = f(x)$. Calculus is used to show that f is maximized at $x = p$, so that $\mathbf{P}^*_{\mathsf{D\text{-}BC}}(n,c) = f(p) = (1-1/(c+1))^c \cdot 1/(c+1)$. The well-known fact that $(1-1/c)^c \approx 1/e$ is not enough to show the precise lower bound $(1-1/(c+1))^c \geq 1/e$; we arrive at it via some Taylor Series estimates. The upper bound on $\mathbf{R}_{\mathsf{D\text{-}BC}}(n,c,m)$ uses a Chernoff Bound.

FIXED-SET-SIZE SAMPLER. Can one do better? It turns out the success probability of $\mathsf{D\text{-}BC}$ is not optimal but still very good; its more important drawback is having non-zero error probability. Our fixed-set-size sampler fills both gaps. Its success probability is optimal, meaning maximal in the class of all HWD samplers, and it achieves this with zero error probability.

For intuition, returning to Eq. (7), let ℓ be such that $\binom{n-c-1}{\ell} \geq \binom{n-c-1}{j}$ for all $j \in [0..n-c-1]$. Then clearly $\mathbf{P}^*_{\mathsf{D}}(n,c)$ is maximized by setting $\mathrm{W}(n,c,c+j) = 1$ if $j = \ell$ and 0 otherwise. That is, all the probability is on strings of Hamming weight $t = c + \ell$. This leads to our fixed-set-size sampler $\mathsf{D\text{-}FX}^t$, which is parameterized by an integer $t \in [c..n-1]$ and shown on the right in Fig. 2. Here $\mathcal{P}(n,t)$ denotes the set of all size t subsets of $[1..n]$. The sampler picks a random set $T \subseteq [1..n]$ of size t and returns as s its characteristic vector, meaning $s[j] = 1$ if $j \in T$ and $s[j] = 0$ otherwise for all $j \in [1..n]$. We let $\mathsf{D\text{-}FX} = \mathsf{D\text{-}FX}^t$ for $t = \lceil (cn-1)/(c+1) \rceil$ and refer to this as the optimal fixed-set-size sampler. Theorem 3 below justifies this name by showing that $\mathsf{D\text{-}FX}$ maximizes the success probability, first across all samplers in the fixed-set-size class, and second across *all HWD samplers*, and meanwhile has error probability is zero.

n	c	D-BC		D-FX		$e(c+1)$
		$1/\mathbf{P}^*_{\mathsf{D\text{-}BC}}(n,c)$	m	$1/\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c)$	m	
100 M	10 K	27, 184	150, 665	27, 183	9, 999	27, 185
100 M	100 K	271, 829	143, 293	271, 694	999	271, 830
100 M	1000 K	2, 718, 283	144, 002	2, 704, 680	99	2, 718, 284
250 M	25 K	67, 958	233, 439	67, 955	9, 999	67, 959
250 M	250 K	679, 571	227, 001	679, 232	999	679, 573
250 M	2500 K	6, 795, 705	228, 633	6, 761, 699	99	6, 795, 707
500 M	50 K	135, 915	327, 085	135, 909	9, 999	135, 916
500 M	500 K	1, 359, 142	321, 695	1, 358, 463	999	1, 359, 143
500 M	5000 K	13, 591, 410	324, 365	13, 523, 397	99	13, 591, 411

Fig. 3. Evaluation of samplers, where n is stated in millions (M) and c in thousands (K). For D-BC we compute m such that $\mathbf{R}_{\mathsf{D\text{-}BC}}(n,c,m)/\mathbf{P}^*_{\mathsf{D\text{-}BC}}(n,c) \le 2^{-128}$ since this will be an additive term in our main theorem (cf. Theorem 4). For D-FX we set $m = \lfloor (n-1)/(c+1) \rfloor$. The last column is our estimate of $e(c+1)$ for the reciprocal of the success probabilities.

Theorem 3. *Let $n, m \ge 1$ and $c, t \ge 0$ be integers such that $n - t \le m$ and $c \le t < n$. Let $\mathsf{D\text{-}FX}^t$ be the fixed-set-size sampler associated to t as per Fig. 2. Then $\mathsf{D\text{-}FX}^t$ is Hamming-weight determined and $\mathbf{P}^*_{\mathsf{D\text{-}FX}^t}(n,c) = \binom{n-c-1}{t-c} \cdot \binom{n}{t}^{-1}$. Furthermore $\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c) \ge \mathbf{P}^*_{\mathsf{D\text{-}FX}^t}(n,c)$ for all $t \in [c..n-1]$. Also $\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c) \ge \mathbf{P}^*_{\mathsf{D}}(n,c)$ for all HWD samplers D and*

$$\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c) \ge \frac{1}{e} \cdot \frac{1}{c+1} \, . \tag{10}$$

Finally, $\mathbf{R}_{\mathsf{D\text{-}FX}}(n,c,m) = 0$.

The proof of Theorem 3 is in the full version [7]. Here we sketch the main ideas. We observe that $\mathsf{D\text{-}FX}^t$ is HWD with Hamming-weight probability function $\mathrm{W}(n,c,\ell) = 1/\binom{n}{\ell}$ if $\ell = t$ and 0 otherwise. Define the function $f : [c..n-1] \to [0,1]$ by $f(t) = \binom{n-c-1}{t-c} \cdot \binom{n}{t}^{-1}$. Then Eq. (7) is used to show that $\mathbf{P}^*_{\mathsf{D\text{-}FX}^t}(n,c) = f(t)$. The function f is too complicated to maximize via calculus. Instead, we look at ratios of consecutive terms, namely we seek t such that $f(t)/f(t+1)$ is as close to 1 as possible. In this way one can show that the maximum of f occurs at $t^* = \lceil (cn-1)/(c+1) \rceil$, which was the choice of t used to define $\mathsf{D\text{-}FX} = \mathsf{D\text{-}FX}^{t^*}$. This shows the claim in the Theorem that $\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c) \ge \mathbf{P}^*_{\mathsf{D\text{-}FX}^t}(n,c)$ for all $t \in [c..n-1]$. The broader optimality claim, namely that $\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c) \ge \mathbf{P}^*_{\mathsf{D}}(n,c)$ for all HWD samplers D, follows from Eq. (7). Equation (10) now follows from Theorem 2. The error probability is clearly zero.

Numerical estimates. Figure 3 gives numbers for a few values of n (in millions) and c (in thousands). We see that $1/\mathbf{P}^*_{\mathsf{D\text{-}FX}}(n,c) \leq 1/\mathbf{P}^*_{\mathsf{D\text{-}BC}}(n,c)$, meaning D-FX is always better than D-BC, but the difference is small. (Intuitively this is because in the biased coin sampler, the expected Hamming weight of the sampled s is $w = cn/(c+1)$, which is very close to $t^* = \lceil w - 1/(c+1) \rceil$.) We see that the approximation $e(c+1)$ is very good, which is why we will use it in our applications. We see that $e(c+1)$, the factor our bounds will give up in applications, is much less than n, the factor that prior bounds gave up. In the least conservative estimation, we set $c = 1\%$ as this already gives improvements for concrete parameters. Depending on the application, an asymptotic improvement can be achieved if c can be bounded by $\sqrt{n}$. For D-BC, we have chosen m to put the error probability at 2^{-128}, and see that it is already significantly less than n, but the $m = \lfloor (n-1)/(c+1) \rfloor$ for D-FX is appreciably better (lower).

Conclusion. Moving forward, we will use the fixed-set-size sampler D-FX and Theorem 3. This may raise the question of why we have considered the biased-coin sampler at all. One reason is that the analysis and results of Theorem 2 are used and needed as comparison points to determine and eventually conclude that Theorem 3 does better. The other reason is historical, namely that the biased-coin sampler is the natural extension of Coron's technique and thus worthy of formulating and analyzing.

4 General Framework and cp-muc Security Theorem

Our technique works across many games (security notions) and schemes. We want to avoid repeating similar proofs each time and also want to understand the scope and limits of the technique: under what conditions does it work, and when does it not work? This Section develops answers to these questions with a general framework.

Our goal is to prove statements of the form "For all security games satisfying a certain condition, mu security can be promoted to cp-muc security." For this to be mathematically sound requires a formal definition of a "security game." The only approach we know, code-based games [10], would require formalizing a programming language. We suggest here a simpler approach suited to our ends. We define an object called a formal security specification (FSS) that, formally, is just an algorithm that functionally specifies the initializations, oracle input-output behaviors (including what happens under corruptions) and final decision of the game underlying the target security notion. To an FSS Π we then associate three (standard, code-based) games capturing su, mu and muc security, respectively. We define locality of an FSS as the condition needed for the result, which is stated and proved in our General Theorem (Theorem 4).

Schemes. An FSS aims to define security of a scheme Sch, so we start with a simple and general abstraction of the latter. Namely, a *scheme* Sch is simply a tuple. Its entries may include algorithms as well as (descriptions of) associated sets or numbers. An example is a signature scheme, which specifies a key-generation algorithm Sch.Kg, a signing algorithm Sch.Sign and a verification algorithm Sch.Vf. As this indicates, we extract individual components of the scheme tuple with dot notation. Another example is an encryption scheme, specifying key-generation algorithm Sch.Kg, encryption algorithm Sch.Enc and decryption algorithm Sch.Dec. It may also specify the length Sch.rl of the randomness (coins) used by the encryption algorithm, and a message space Sch.MS.

In the ROM, it is often the case that the range set of the RO needed depends on the scheme. (For example, for a KEM, it could be the set of strings of length the desired session key.) To accommodate this, we allow schemes to name the set Sch.ROS from which they ask their random oracle to be drawn.

```
Game $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$
INIT:
  $\mathsf{h} \leftarrow\!\!\$\ \mathsf{Sch.ROS}$ ; $(pp, os) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\texttt{gs})$
  $(\mathrm{iout}, \mathrm{St}) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\texttt{init}, (pp, os))$ ; Return $(pp, \mathrm{iout})$
ORACLE($\texttt{name}$, arg): // $\texttt{name} \notin \{\texttt{gs}, \texttt{init}, \texttt{fin}, \texttt{corr}\}$
  $(\mathrm{oout}, \mathrm{St}) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\texttt{name}, \mathrm{arg}, \mathrm{St})$ ; Return oout
RO($x$): // Random oracle
  $y \leftarrow \mathsf{h}(x)$ ; Return $y$
FIN(farg):
  $\mathrm{dec} \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\texttt{fin}, \mathrm{farg}, \mathrm{St})$ ; Return dec
```

```
$\mathsf{UF}[\mathsf{Sig}, \mathsf{h}](\texttt{gs})$:
  Return $(\varepsilon, \varepsilon)$
$\mathsf{UF}[\mathsf{Sig}, \mathsf{h}](\texttt{init}, (\varepsilon, \varepsilon))$:
  $(vk, sk) \leftarrow\!\!\$\ \mathsf{Sig.Kg}[\mathsf{h}]$
  $\mathrm{St.vk} \leftarrow vk$ ; $\mathrm{St.sk} \leftarrow sk$ ; $\mathrm{St.S} \leftarrow \emptyset$
  Return $(vk, \mathrm{St})$
$\mathsf{UF}[\mathsf{Sig}, \mathsf{h}](\texttt{sign}, M, \mathrm{St})$:
  $\sigma \leftarrow\!\!\$\ \mathsf{Sig.Sign}[\mathsf{h}](\mathrm{St.sk}, M)$
  $\mathrm{St.S} \leftarrow \mathrm{St.S} \cup \{M\}$ ; Return $(\sigma, \mathrm{St})$
$\mathsf{UF}[\mathsf{Sig}, \mathsf{h}](\texttt{corr}, \varepsilon, \mathrm{St})$:
  Return $(\mathrm{St.sk}, \mathrm{St})$
$\mathsf{UF}[\mathsf{Sig}, \mathsf{h}](\texttt{fin}, (M, \sigma), \mathrm{St})$:
  If $M \in \mathrm{St.S}$ then return false
  Return $\mathsf{Sig.Vf}[\mathsf{h}](\mathrm{St.vk}, M, \sigma)$
```

```
Game $\mathbf{G}^{\mathsf{UF}}_{\mathsf{Sig}}$
INIT:
  $\mathsf{h} \leftarrow\!\!\$\ \mathsf{Sig.ROS}$ ; $\mathrm{S} \leftarrow \emptyset$
  $(vk, sk) \leftarrow\!\!\$\ \mathsf{Sig.Kg}[\mathsf{h}]$
  Return $vk$
ORACLE($\texttt{sign}$, $M$):
  $\sigma \leftarrow\!\!\$\ \mathsf{Sig.Sign}[\mathsf{h}](sk, M)$
  $\mathrm{S} \leftarrow \mathrm{S} \cup \{M\}$
  Return $\sigma$
RO($x$): // Random oracle
  $y \leftarrow \mathsf{h}(x)$ ; Return $y$
FIN($M, \sigma$):
  If $M \in \mathrm{S}$ then return false
  Return $\mathsf{Sig.Vf}[\mathsf{h}](vk, M, \sigma)$
```

Fig. 4. Top: Game associated to formal security specification Π and scheme Sch in the single-user setting. **Bottom Left:** Formal security specification UF. **Bottom Right:** The $\mathbf{G}^{\mathsf{UF}}_{\mathsf{Sig}}$ game, rewritten in standard form.

FORMAL SECURITY SPECIFICATIONS AND THE SINGLE-USER GAME. A *formal security specification* (FSS) Π is an algorithm whose input is a string $\texttt{name}$, drawn from a finite set $\mathsf{Names} \subseteq \{0,1\}^*$ associated to Π, that serves to name a sub-algorithm $\Pi(\texttt{name}, \cdots)$. The names $\texttt{gs}, \texttt{init}, \texttt{fin} \in \mathsf{Names}$, and in our context also $\texttt{corr} \in \mathsf{Names}$, are reserved; they stand, respectively, for "Global Setup," "Initialize," "Finalize" and "Corrupt." Particular FSSs can define more names. Through (code-based) game $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$ shown in Fig. 4, Π specifies a notion of security for a scheme Sch. In INIT, line 1 picks, from the random oracle space of the scheme, a function h that will serve as the random oracle. The global setup sub-algorithm $\Pi(\texttt{gs})$ of the FSS is then run to produce a pair $(pp, os) \leftarrow\$ \Pi[\mathsf{Sch}, \mathsf{h}](\texttt{gs})$ whose components are called the public parameters and oracle secrets, respectively. (An example os is a global challenge bit for an ind-style game.) As the notation indicates, sub-algorithms of Π have access to the scheme Sch and may thus call its algorithms, and they also have oracle access to h. At line 2, the initialize sub-algorithm $\Pi(\texttt{init}, \cdot)$ is run. It takes as input pp, os and produces an output iout together with an initial state St. The adversary is given (pp, iout). The $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$ game will maintain the state St and update it as necessary. After that, the FSS maps to the actual game in a straightforward way. For any $\texttt{name} \in \{\texttt{gs}, \texttt{init}, \texttt{fin}, \texttt{corr}\}$, sub-algorithm $\Pi(\texttt{name}, \cdots)$ implements an actual oracle $\text{ORACLE}(\texttt{name}, \cdots)$ in $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$. Given an argument arg, the latter runs $\Pi[\mathsf{Sch}, \mathsf{h}](\texttt{name}, \text{arg}, \text{St})$, where St is the current state, to get an output oout (that is returned to the adversary) and updated state St (that is maintained by game $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$). Game $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$ also provides a random oracle RO that gives the adversary access to h. FIN uses sub-algorithm $\Pi(\texttt{fin}, \cdots)$ to return a game decision dec. Beyond the argument farg provided by the adversary, $\Pi(\texttt{fin}, \cdots)$ gets the current state as input.

As an example, we show on the bottom left of Fig. 4 the FSS UF corresponding to uf-security of a signature scheme Sig. To its right we show $\mathbf{G}^{\mathsf{UF}}_{\mathsf{Sig}}$ re-written in the usual way to see that it is the standard game. Of course, if one is interested in just one notion (such as uf-security) one can give $\mathbf{G}^{\mathsf{UF}}_{\mathsf{Sig}}$ directly and there is no need for FSSs, but as we have discussed, FSSs will allow us to give precise yet general results covering many games, and also allow us to classify games into different types.

The $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$ game captures the single-user setting and does not use sub-algorithm $\Pi(\texttt{corr}, \cdots)$; it will be used below. The difference between pp and os is that the FSS gets both but the adversary gets only the former. This distinction will allow us to distill exactly what our general result needs to work and also let us to apply the latter broadly and in settings where "corrupt" does not have the usual interpretation.

An FSS Π has an associated type, which is either "search" or "decision." The advantage of an adversary $\mathcal{A}$ playing $\mathbf{G}^{\Pi}_{\mathsf{Sch}}$ is defined as $\mathbf{Adv}^{\Pi}_{\mathsf{Sch}}(\mathcal{A}) = \Pr[\mathbf{G}^{\Pi}_{\mathsf{Sch}}(\mathcal{A})]$ if Π is a search FSS and $\mathbf{Adv}^{\Pi}_{\mathsf{Sch}}(\mathcal{A}) = 2\Pr[\mathbf{G}^{\Pi}_{\mathsf{Sch}}(\mathcal{A})] - 1$ if Π is a decision FSS. In the latter case we will make an extra technical assumption, namely that for all $\mathsf{Sch}, \mathsf{h}, \text{St}$ the decision $\text{dec} \leftarrow\$ \Pi[\mathsf{Sch}, \mathsf{h}](\texttt{fin}, \text{farg}, \text{St})$ is uniformly distributed over $\{\mathsf{true}, \mathsf{false}\}$ when $\text{farg} \leftarrow\$ \{0,1\}$. This is typically true because os is a random challenge bit and $\Pi[\mathsf{Sch}, \mathsf{h}](\texttt{fin}, \text{farg}, \text{St})$ returns

```
Games $\mathbf{G}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}$ and $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$
INIT:
  $\mathsf{h} \leftarrow\!\!\$\ \mathsf{Sch.ROS}$ ; $(pp, os) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{gs})$
  For $i = 1, \ldots, n$ do $(\mathbf{iout}[i], \mathbf{St}[i]) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{init}, (pp, os))$
  Return $(pp, \mathbf{iout})$
ORACLE$(\mathtt{name}, (i, \mathrm{arg}))$: // $\mathtt{name} \notin \{\mathtt{gs}, \mathtt{init}, \mathtt{fin}, \mathtt{corr}\}$ and $i \in [1..n]$
  $(\mathrm{oout}, \mathbf{St}[i]) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{name}, \mathrm{arg}, \mathbf{St}[i])$ ; Return oout
CORRUPT$(i, \mathrm{arg})$: // Game $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$ only
  $\mathrm{CS} \leftarrow \mathrm{CS} \cup \{i\}$
  If $|\mathrm{CS}| > c$ then return $\bot$
  $(\mathrm{oout}, \mathbf{St}[i]) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{corr}, \mathrm{arg}, \mathbf{St}[i])$ ; Return oout
RO$(x)$: // Random oracle
  $y \leftarrow \mathsf{h}(x)$ ; Return $y$
FIN$(i, \mathrm{farg})$: // $i \in [1..n]$
  If $i \in \mathrm{CS}$ then return $\mathsf{false}$
  $\mathrm{dec} \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{fin}, \mathrm{farg}, \mathbf{St}[i])$ ; Return dec
```

Fig. 5. Game describing the execution of a formal security specification Π with scheme Sch in the multi-user setting, both without corruptions (mu) and with corruptions (muc). The difference is that oracle CORRUPT is included only in the second game.

$[[\mathrm{farg} = os]]$, but it does not in general follow merely through the definition of the advantage as $2\Pr[\mathbf{G}^{\Pi}_{\mathsf{Sch}}(\mathcal{A})] - 1$. This will be used in the proof of Theorem 4.

We say that an FSS Π is *local* if the oracle secret os is always ε. (A bit more formally, the probability that $os = \varepsilon$ when $(pp, os) \leftarrow\!\!\$\ \Pi[\mathsf{Sch}, \mathsf{h}](\mathtt{gs})$ is 1 for all Sch, h.) Intuitively, different users in (upcoming) games $\mathbf{G}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}$ and $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$ will have no secret common to them all but unknown to the adversary. This captures the condition for our general result of Theorem 4.

MULTI-USER SECURITY FOR AN FSS. To FSS Π, scheme Sch, a number $n \geq 1$ of users and a number $c \geq 0$ of corruptions, we now associate a multi-user (without corruptions) game $\mathbf{G}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}$ and a multi-user with corruptions game $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$ as shown in Fig. 5. We see here the reason to separate initialization into the two sub-algorithms $\Pi(\mathtt{gs})$ and $\Pi(\mathtt{init}, \cdots)$: The first is run once to produce parameters common to all users, while the second is run once per user and produces a state $\mathbf{St}[i]$ for each user $i \in [1..n]$. Oracle ORACLE$(\mathtt{name}, \cdots)$ now takes a user identity i in addition to arg and answers via the sub-algorithm $\Pi(\mathtt{name}, \cdots)$ using the state of user i. FIN also takes a user identity i and returns its decision using the state of that user. Oracle CORRUPT is present only in game $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$ and answers as per the associated sub-algorithm, again for a given user. The advantage $\mathbf{Adv}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}(\mathcal{A})$ of an adversary $\mathcal{A}$ playing $\mathbf{G}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}$ is defined as $\Pr[\mathbf{G}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}(\mathcal{A})]$ if Π is a search FSS and $2\Pr[\mathbf{G}^{\Pi\text{-mu-}n}_{\mathsf{Sch}}(\mathcal{A})] - 1$ if Π is a decision FSS, and correspondingly $\mathbf{Adv}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}(\mathcal{A})$ is defined as $\Pr[\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}(\mathcal{A})]$ or $2\Pr[\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}(\mathcal{A})] - 1$.

General cp-muc theorems. We give our main, general results that promote mu to corruption-parameterized muc security for *all* local game specifications. Later we will see many applications. Below we crucially leverage HWD samplers.

Theorem 4. *Let $n, m \geq 1$ and $c \geq 0$ be integers such that $m \leq n$ and $c < n$. Let D be a Hamming-weight determined sampler. Let $\alpha = \mathbf{P}^*_{\mathsf{D}}(n, c)$ and $\beta = \mathbf{R}_{\mathsf{D}}(n, c, m)$. Let Sch be a scheme, and Π a formal security specification for it. Assume Π is local. Let $\gamma = 1$ if Π is a search-type* FSS *and $\gamma = 2$ if Π is a decision-type* FSS. *Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}$ such that*

$$\mathbf{Adv}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}(\mathcal{A}) \leq (1/\alpha) \cdot \mathbf{Adv}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}(\mathcal{B}) + \gamma \cdot \beta/\alpha \ . \tag{11}$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler D.

Before discussing the proof we state a corollary obtained by plugging the fixed-set-size sampler into the above; this is what we will most often use in applications.

Theorem 5. *Let n, c be integers such that $0 \leq c < n$, and let $m = \lfloor (n-1)/(c+1) \rfloor$. Let Sch be a scheme, and Π a formal security specification for it. Assume Π is local. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}$ such that*

$$\mathbf{Adv}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}(\mathcal{B}) \ . \tag{12}$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler $\mathsf{D\text{-}FX}$.

Proof (Theorem 5). Let $\mathsf{D} = \mathsf{D\text{-}FX}$ be the optimal fixed-set-size sampler and apply Theorem 4. We have $\alpha = \mathbf{P}^*_{\mathsf{D\text{-}FX}}(n, c)$ and $\beta = \mathbf{R}_{\mathsf{D\text{-}FX}}(n, c, m)$ in Eq. (11). Now Theorem 3 says that $1/\alpha \leq e(c+1)$, while $\beta = 0$, yielding Eq. (12). □

It remains to prove Theorem 4. A full proof is in the full version [7]. Here we give an overview.

Adversary $\mathcal{B}$ (shown in detail in the full version [7]) picks s by running the sampler D, thereby coloring each user $i \in [1..n]$ as either red ($s[i] = 1$) or blue ($s[i] = 0$). It now runs $\mathcal{A}$. In answering the game-$\mathbf{G}^{\Pi\text{-muc-}(n,c)}_{\mathsf{Sch}}$ queries that $\mathcal{A}$ makes, $\mathcal{B}$ will simulate a red user i directly. This means it will pick and maintain this user's state $\mathbf{St}[i]$, allowing it to answer all queries to this user, including, most importantly, a Corrupt(i) query. Meanwhile, it will aim to identify blue users with the users in its own underlying $\mathbf{G}^{\Pi\text{-mu-}m}_{\mathsf{Sch}}$ game, using its oracles from that game to reply to queries of $\mathcal{A}$ for these users. It hopes that $\mathcal{A}$ runs to a win with a Fin query to a blue user, in which case $\mathcal{B}$ will win its own game.

There are a number of bad events, ways in which this strategy can fail. The first is that $n - \mathbf{w}_{\mathrm{H}}(s) > m$, meaning the number of blue users is too large for them to be mapped to the users in game $\mathbf{G}_{\mathsf{Sch}}^{\Pi\text{-mu-}m}$. Letting G_0 be a game capturing $\mathcal{B}$'s advantage, a first game hop will bound the probability of this bad event via the First Fundamental Lemma of Game Playing (Lemma 1), and lead to the β/α term in the bound while putting us now in a game G_1 where the limitation on the number of blue users has vanished. A first game hop will bound the probability of this event via the First Fundamental Lemma of Game Playing, leading to the $c\beta$ term in the bound and putting us now in a game where the limitation on the number of blue users has vanished. The second source of failure is that either there is a CORRUPT query to a blue user ($\mathcal{B}$ has no way to answer this) or $\mathcal{A}$'s FIN query is to a red user (in which case $\mathcal{B}$ won't win). Game G_2 flags this through settings of flag bad. The difficulty is that the probability (that bad is set) is large, namely close to 1, and not small. So, while handling it again via the First Fundamental Lemma of Game Playing is possible, it would yield a large additive term in the bound, making the latter vacuous. Instead, letting GD be the probability that bad is not set and W the event that $\mathcal{B}$ wins, we seek to lower bound $\Pr[\mathrm{W} \wedge \mathrm{GD}]$ in G_2. A crucial use of the *Second* Fundamental Lemma of Game Playing (Lemma 2) equates this with the same probability in a game G_3 where events GD and W are (unlike in G_2) independent. This leaves us with the product $\Pr[\mathrm{W}]\cdot\Pr[\mathrm{GD}]$ in G_3. To conclude, we will lower bound $\Pr[\mathrm{W}]$ in terms of the advantage of $\mathcal{A}$. Then we will use Theorem 1 to lower bound $\Pr[\mathrm{GD}]$ in terms of the sampler success probability $\mathbf{P}_{\mathsf{D}}^{*}(n, c)$ of Eq. Equation (7).

A wrinkle is that the analysis sketched above is for the case that Π is a search-type FSS. The case of it being a decision-type FSS is more delicate. We will need to ensure that when bad events happen, $\mathcal{B}$ has advantage zero, which is done leveraging the assumption we have made that decision formal security specifications return a random decision in this case. While the adversary $\mathcal{B}$ and the games will be written to cover both cases (search and decision), the analyses are different enough that, below, we give them separately. (And indeed the bound in Eq. (11) is slightly different.)

Why do we need to assume that Π is local? The secret os in the $\mathbf{G}_{\mathsf{Sch}}^{\Pi\text{-mu-}n}$ game is not available to $\mathcal{B}$, yet it has to simulate game $\mathbf{G}_{\mathsf{Sch}}^{\Pi\text{-muc-}(n,c)}$ for $\mathcal{A}$ in such a way that it is underlain by this same os. This happens correctly for blue users because their queries are forwarded, but $\mathcal{B}$ cannot simulate the oracle replies for red users to be consistent with os. If the FSS is local, however, $os = \varepsilon$ so the difficulty goes away.

5 Applications

We can promote mu to cp-muc for many target goals and schemes simply by noting that the FSS is local and applying Theorem 5. We call these direct applications, and below we illustrate with a few examples, but there are many more. However, the FSSs for some important goals, most notably ind-style games with a single challenge bit across all users, are not local. Nonetheless, we can give dedicated proofs for specific schemes. Since these results use Theorem 5 in an intermediate step, we call them indirect applications.

5.1 Direct Applications

Digital Signatures. We formally state our theorem for signature schemes underlying the informal claim of Eq. (2) in our use of signatures as an example in the Introduction. The FSS for signatures is UF (Fig. 4), which is local. The corresponding mu(c) games are obtained via UF as described in Fig. 5. Theorem 5 now yields:

Theorem 6. *Let n, c be integers such that $0 \leq c < n$, and let $m = \lfloor (n-1)/(c+1) \rfloor$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}_{\mathsf{Sig}}^{\mathsf{UF}\text{-muc-}(n,c)}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}_{\mathsf{Sig}}^{\mathsf{UF}\text{-mu-}m}$ such that*

$$\mathbf{Adv}_{\mathsf{Sig}}^{\mathsf{UF}\text{-muc-}(n,c)}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}_{\mathsf{Sig}}^{\mathsf{UF}\text{-mu-}m}(\mathcal{B}) .$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler D-FX.

As explained in Sect. 1.1: (1) for tightly mu secure schemes, this implies $\mathbf{Adv}_{\mathsf{Sig}}^{\mathsf{UF}\text{-muc-}(n,c)}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}_{\mathsf{Sig}}^{\mathsf{UF}}(\mathcal{B})$, a significant improvement over the factor n from the hybrid argument bound, and (2) this yields tightness improvements for standardized schemes that are used in practical applications such as AKE, which we will discuss in Sect. 5.2. Further, we note that the analogous result is true for strongly-unforgeability of signatures.

Public-Key Encryption. Mu (and thus muc) security for public-key encryption and KEMs can be defined with a single challenge bit across all users [4], which we call the single-bit (sb) setting, or with a per-user challenge bit, which we call the multi-bit (mb) setting. The settings are asymptotically equivalent, but we are interested in tightness. Heum and Stam [43] show that without corruptions, SB security is the stronger definition, but which is stronger in the presence of adaptive corruptions is open.

Our framework is able to capture both settings, in the sense that we can give FSSs for both, but interestingly, only the multi-bit one is local, so our general result only applies in the MB setting. We discuss this here; we give results for the SB setting in Sect. 5.2.

Our FSS CCA-MB for KEMs capturing indistinguishability under chosen-ciphertext attacks in the MB setting is shown in Fig. 6. For completeness, we give the muc game in the full version [7]. It picks one challenge bit for each user which is used for encryption queries. During the game, the adversary can adaptively learn user's decryption keys via a corruption, thus also learning the user's challenge bit. In the end, the adversary has to guess the challenge bit of an uncorrupted user. Note that CCA-MB does not need oracle secret, i. e., it is local, and we can apply Theorem 5 to obtain the following result.

```
$\underline{\mathsf{CCA\text{-}MB}[\mathsf{KEM},\mathsf{h}](\mathtt{gs})\text{:}}$
  Return $(\varepsilon,\varepsilon)$
$\underline{\mathsf{CCA\text{-}MB}[\mathsf{KEM},\mathsf{h}](\mathtt{init},\varepsilon)\text{:}}$
  $(ek,dk) \leftarrow\!\$\ \mathsf{KEM.Kg}[\mathsf{h}]$
  $\mathrm{St.ek} \leftarrow ek$ ; $\mathrm{St.dk} \leftarrow dk$
  $b \leftarrow\!\$\ \{0,1\}$ ; $\mathrm{St.b} \leftarrow b$
  Return $(ek,\mathrm{St})$
$\underline{\mathsf{CCA\text{-}MB}[\mathsf{KEM},\mathsf{h}](\mathtt{enc},\varepsilon,\mathrm{St})\text{:}}$
  $(C,K_0) \leftarrow\!\$\ \mathsf{KEM.Encaps}[\mathsf{h}](\mathrm{St.ek})$
  $K_1 \leftarrow\!\$\ \{0,1\}^{\mathsf{KEM.kl}}$
  $\mathrm{St.S} \leftarrow \mathrm{St.S} \cup \{C\}$
  Return $(C, K_{\mathrm{St.b}}, \mathrm{St})$
$\underline{\mathsf{CCA\text{-}MB}[\mathsf{KEM},\mathsf{h}](\mathtt{dec},C,\mathrm{St})\text{:}}$
  If $C \in \mathrm{St.S}$ then return $\perp$
  $K \leftarrow \mathsf{KEM.Decaps}[\mathsf{h}](\mathrm{St.dk},C)$
  Return $(K,\mathrm{St})$
$\underline{\mathsf{CCA\text{-}MB}[\mathsf{KEM},\mathsf{h}](\mathtt{corr},\varepsilon,\mathrm{St})\text{:}}$
  Return $(\mathrm{St.dk},\mathrm{St})$
$\underline{\mathsf{CCA\text{-}MB}[\mathsf{KEM},\mathsf{h}](\mathtt{fin},b',\mathrm{St})\text{:}}$
  Return $([\![\mathrm{St.b} = b']\!],\mathrm{St})$
```

```
$\underline{\mathsf{CCA\text{-}SB}[\mathsf{KEM},\mathsf{h}](\mathtt{gs})\text{:}}$
  $b \leftarrow\!\$\ \{0,1\}$ ; Return $(\varepsilon,b)$
$\underline{\mathsf{CCA\text{-}SB}[\mathsf{KEM},\mathsf{h}](\mathtt{init},(\varepsilon,b))\text{:}}$
  $(ek,dk) \leftarrow\!\$\ \mathsf{KEM.Kg}[\mathsf{h}]$
  $\mathrm{St.ek} \leftarrow ek$ ; $\mathrm{St.dk} \leftarrow dk$
  $\mathrm{St.corr} \leftarrow \mathsf{false}$ ; $\mathrm{St.b} \leftarrow b$
  Return $(ek,\mathrm{St})$
$\underline{\mathsf{CCA\text{-}SB}[\mathsf{KEM},\mathsf{h}](\mathtt{enc},\varepsilon,\mathrm{St})\text{:}}$
  If St.corr then return $\perp$
  $(C,K_0) \leftarrow\!\$\ \mathsf{KEM.Encaps}[\mathsf{h}](\mathrm{St.ek})$
  $K_1 \leftarrow\!\$\ \{0,1\}^{\mathsf{KEM.kl}}$ ; $\mathrm{St.S} \leftarrow \mathrm{St.S} \cup \{C\}$
  Return $((C, K_{\mathrm{St.b}}), \mathrm{St})$
$\underline{\mathsf{CCA\text{-}SB}[\mathsf{KEM},\mathsf{h}](\mathtt{dec},C,\mathrm{St})\text{:}}$
  If $C \in \mathrm{St.S}$ then return $\perp$
  $K \leftarrow \mathsf{KEM.Decaps}[\mathsf{h}](\mathrm{St.dk},C)$
  Return $(K,\mathrm{St})$
$\underline{\mathsf{CCA\text{-}SB}[\mathsf{KEM},\mathsf{h}](\mathtt{corr},\varepsilon,\mathrm{St})\text{:}}$
  If $\mathrm{St.S} \neq \emptyset$ then return $\perp$
  $\mathrm{St.corr} \leftarrow \mathsf{true}$ ; Return $(\mathrm{St.dk},\mathrm{St})$
$\underline{\mathsf{CCA\text{-}SB}[\mathsf{KEM},\mathsf{h}](\mathtt{fin},b',\mathrm{St})\text{:}}$
  Return $([\![\mathrm{St.b} = b']\!],\mathrm{St})$
```

Fig. 6. Left: Formal security specification $\mathsf{CCA\text{-}MB}[\mathsf{KEM},\mathsf{h}]$ capturing security with multiple challenge bits. **Right:** Formal security specification $\mathsf{CCA\text{-}SB}[\mathsf{KEM},\mathsf{h}]$ capturing security with a single challenge bit.

Theorem 7. *Let n, c be integers such that $0 \le c < n$, and let $m = \lfloor (n-1)/(c+1) \rfloor$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\mathsf{CCA\text{-}MB}\text{-muc-}(n,c)}_{\mathsf{KEM}}$. Then we construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\mathsf{CCA\text{-}MB}\text{-mu-}m}_{\mathsf{KEM}}$ such that*

$$\mathbf{Adv}^{\mathsf{CCA\text{-}MB}\text{-muc-}(n,c)}_{\mathsf{KEM}}(\mathcal{A}) \le e(c+1) \cdot \mathbf{Adv}^{\mathsf{CCA\text{-}MB}\text{-mu-}m}_{\mathsf{KEM}}(\mathcal{B}) \ .$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler $\mathsf{D\text{-}FX}$.

As a second illustrative example, we consider one-way security of PKE under plaintext checking and ciphertext validity attacks [21,44], which serves as a useful intermediate notion for the Fujisaki-Okamoto (FO) transform [26,27], which we will have a closer look at in Sect. 5.2. The FSS $\mathsf{OW\text{-}PCVA}$ is given in the full version [7]. It is local, and thus we get:

Theorem 8. *Let n, c be integers such that $0 \le c < n$, and let $m = \lfloor (n-1)/(c+1) \rfloor$. Let $\mathcal{A}$ be an adversary for game $\mathbf{G}^{\mathsf{OW\text{-}PCVA}\text{-muc-}(n,c)}_{\mathsf{PKE}}$. Then we construct an* *adversary $\mathcal{B}$ for game* $\mathbf{G}_{\mathsf{PKE}}^{\mathsf{OW\text{-}PCVA\text{-}mu\text{-}}m}$ *such that*

$$\mathbf{Adv}_{\mathsf{PKE}}^{\mathsf{OW\text{-}PCVA\text{-}muc\text{-}}n}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}_{\mathsf{PKE}}^{\mathsf{OW\text{-}PCVA\text{-}mu\text{-}}m}(\mathcal{B}) \ .$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler D-FX.

<u>Authenticated Encryption.</u> A (nonce-based) symmetric encryption (SE) scheme SE specifies deterministic algorithms SE.Enc and SE.Dec, a key length SE.kl and nonce space $\mathcal{N}$. The encryption algorithm takes as input a key $k \in \{0,1\}^{\mathsf{SE.kl}}$, a nonce $N \in \mathcal{N}$ and a message $M \in \{0,1\}^*$, and returns a ciphertext $C \in \{0,1\}^{\mathsf{SE.cl}(|M|)}$, where SE.cl is the ciphertext length function. The decryption algorithm takes as input k, N, C and returns either a message $M \in \{0,1\}^*$ or $\perp$ indicating failure. Correctness asks that for every $k \in \{0,1\}^{\mathsf{SE.kl}}$, $N \in \mathcal{N}$, $M \in \{0,1\}^*$, if $C \leftarrow \mathsf{SE.Enc}(k, N, M)$, then $\mathsf{SE.Dec}(k, N, C)$ returns M.

Multi-user security with corruptions for authenticated encryption was studied in [48], in both the single and the multi-bit setting. As for PKE schemes, only the multi-bit setting can be expressed via a local FSS which we provide in the full version [7]. We then apply our general theorem to get the following.

Theorem 9. *Let n, c be integers such that $0 \leq c < n$, and let $m = \lfloor (n-1)/(c+1) \rfloor$. Let* SE *be an SE scheme and let $\mathcal{A}$ be an adversary for game* $\mathbf{G}_{\mathsf{SE}}^{\mathsf{AE\text{-}MB\text{-}muc\text{-}}(n,c)}$. *Then we construct an adversary $\mathcal{B}$ for game* $\mathbf{G}_{\mathsf{SE}}^{\mathsf{AE\text{-}MB\text{-}mu\text{-}}m}$ *such that*

$$\mathbf{Adv}_{\mathsf{SE}}^{\mathsf{AE\text{-}MB\text{-}muc\text{-}}(n,c)}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}_{\mathsf{SE}}^{\mathsf{AE\text{-}MB\text{-}mu\text{-}}m}(\mathcal{B}) \ .$$

Adversary $\mathcal{B}$ makes, to any oracle in its game, the same number of queries as $\mathcal{A}$ made. The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler D-FX.

5.2 Indirect Applications

Our general result does not apply to FSS which are not local, i. e., the game depends on some global oracle secret like a single challenge bit which is used across multiple users. This motivates us to study settings where the target game (and possibly the starting game) are described by non-local FSS, but where we can still achieve muc-security with a tightness loss of c via some transformation.

Here, we are also interested in settings where corruptions are not necessarily referring to signing or decryption keys. As an example, we show how to apply our framework to recover Coron's result on RSA-FDH signatures in the full version [7]. We now provide more details for our new results.

<u>SB security for KEMs.</u> Bellare, Boldyreva and Micali [4] introduced SB-mu security and also showed that su security implies mu security with a loss linear in the number of users. This initiated further study on tightly-secure PKE and KEM schemes in the mu setting [28,29,39,45,55]. Recently, we have also seen advances on tightly-secure schemes in the muc setting [38,40,54]. As for signature

schemes, we can however observe that the latter schemes are less efficient in terms of size (of public keys and ciphertexts) and computation complexity than canonical schemes.

In Fig. 6, we define the FSS $\mathsf{CCA\text{-}SB}$ for KEM schemes capturing indistinguishability under chosen-ciphertext attacks in the SB setting. (The one for PKE is similar and provided in the full version [7].) The game setup chooses a global challenge bit (stored as oracle secret os); thus the game is not local. This also requires to restrict queries to the challenge and corruption oracle, otherwise the adversary can trivially learn the challenge bit.

The main motivation to study security in the presence of adaptive corruptions for KEMs is that they are of high importance in practice. Here, the SB setting is particularly interesting for composition using the KEM/DEM paradigm [18]. While it has already been studied in the mu setting without corruptions [31,69], the result can be trivially extended to the muc setting using a $\mathsf{CCA\text{-}SB\text{-}muc}$ secure KEM. (This composition result was recently studied for simulation-based security definitions [46].) For these reasons, we now aim to establish $\mathsf{CCA\text{-}SB\text{-}muc}$ security with improved tightness for practical encryption schemes.

Hashed ElGamal. On the left-hand side of Fig. 7, we describe the KEM underlying the hashed ElGamal encryption scheme, which we denote by HEG. It is known that HEG can be proven tightly mu secure under the strong computational Diffie-Hellman ($\mathsf{St\text{-}CDH}$) assumption [1] which we define in terms of game $\mathbf{G}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}$ on the right-hand side of Fig. 7. We denote the advantage of an adversary $\mathcal{A}$ in the game by $\mathbf{Adv}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}(\mathcal{A})$. For muc security, we only know the generic result with a security loss in the number of users. Thus, we ask whether we can do better using our technique and we provide an affirmative answer.

```
HEG.Kg:
  1 $x \leftarrow_\$ \mathbb{Z}_p$ ; $X \leftarrow g^x$
  2 Return $(ek, dk) \leftarrow (X, (X, x))$
HEG.Encaps[H]($ek$):
  3 $r \leftarrow_\$ \mathbb{Z}_p$; $C \leftarrow g^r$ ; $K \leftarrow \mathsf{H}(ek, C, ek^r)$
  4 Return $(C, K)$
HEG.Decaps[H]($dk, C$):
  5 $(ek, x) \leftarrow dk$ ; Return $K \leftarrow \mathsf{H}(ek, C, C^x)$
```

```
$\mathbf{G}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}$
INIT:
  1 $x, y \leftarrow_\$ \mathbb{Z}_p$
  2 Return $(g^x, g^y)$
DDH($Y, Z$):
  3 Return $[\![Z = Y^x]\!]$
FIN($Z$):
  4 Return $[\![Z = g^{xy}]\!]$
```

Fig. 7. Left: Hashed ElGamal KEM HEG for a group $(\mathbb{G}, p, g)$, where $\mathsf{HEG.ROS}$ is the set of all $\mathsf{H} : \{0,1\}^* \rightarrow \{0,1\}^{\mathsf{HEG.kl}}$. **Right:** Game for the strong computational Diffie-Hellman assumption in group $(\mathbb{G}, p, g)$.

Theorem 10. *Let* HEG *be the KEM defined in Fig. 7. Let* n, c *be integers such that* $0 \le c < n$. *Let* $\mathcal{A}$ *be an adversary for game* $\mathbf{G}^{\mathsf{CCA\text{-}SB\text{-}muc\text{-}}(n,c)}_{\mathsf{HEG}}$. *Then we* *construct an adversary $\mathcal{B}$ for game $\mathbf{G}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}$ such that*

$$\mathbf{Adv}^{\mathsf{CCA\text{-}SB}\text{-muc-}(n,c)}_{\mathsf{HEG}}(\mathcal{A}) \leq e(c+1) \cdot \mathbf{Adv}^{\mathsf{St\text{-}CDH}}_{(\mathbb{G},p,g)}(\mathcal{B}) \ .$$

Let q_{ro} be the number of random oracle queries $\mathcal{A}$ makes. Then $\mathcal{B}$ makes at most q_{ro} queries to oracle Ddh. *The running time of $\mathcal{B}$ is about that of $\mathcal{A}$ plus the time for an execution of the sampler* D-FX, *re-randomization of group elements and maintaining lists of encryption, decryption and random oracle queries.*

The proof of the theorem can be found in the full version [7], where we use a multi-user variant of St-CDH which we can write as a local FSS.

Fujisaki-Okamoto transform. The Fujisaki-Okamoto (FO) transform [26, 27] is a generic approach that turns any cpa secure PKE scheme into one that is cca secure in the random oracle model. (Here, we define CPA-SB in terms of CCA-SB, by restricting attention to adversaries that do not query the decryption oracle.) Especially since the announcement of NIST's post-quantum competition [60], the FO transform has seen adoption in the design of many schemes for practical use, including Kyber [15] which has been selected for standardization. The FO has been studied in the mu setting [25], showing that CPA-SB-mu security of the PKE scheme tightly implies CCA-SB-mu security of the transformed KEM.

```
$\overline{\mathsf{PKE}}.\mathsf{Enc}[\mathsf{G}](ek, M)$:
  $C \leftarrow\!\$\ \mathsf{PKE.Enc}(ek, M; \mathsf{G}(ek, M))$
  Return $C$

$\overline{\mathsf{PKE}}.\mathsf{Dec}[\mathsf{G}](dk, c)$:
  $M \leftarrow \mathsf{PKE.Dec}(dk, C)$
  If $M = \bot$ or $\mathsf{PKE.Enc}(ek, M; \mathsf{G}(ek, M)) \neq C$
    Return $\bot$
  Return $M$
```

```
$\mathsf{KEM.Encaps}[\mathsf{G},\mathsf{H}](ek)$:
  $M \leftarrow\!\$\ \overline{\mathsf{PKE}}.\mathsf{MS}$
  $C \leftarrow\!\$\ \overline{\mathsf{PKE}}.\mathsf{Enc}[\mathsf{G}](ek, M)$
  $K \leftarrow \mathsf{H}(ek, M, C)$ ; Return $(C, K)$

$\mathsf{KEM.Decaps}[\mathsf{G},\mathsf{H}](dk, C)$:
  $M \leftarrow \overline{\mathsf{PKE}}.\mathsf{Dec}[\mathsf{G}](dk, C)$
  If $M = \bot$ then return $\bot$
  Return $K \leftarrow \mathsf{H}(ek, M, C)$
```

Fig. 8. Left: PKE scheme $\overline{\mathsf{PKE}} = \mathbf{T}[\mathsf{PKE}]$ obtained from the T-Transform, where $\overline{\mathsf{PKE}}.\mathsf{ROS}$ is the set of all $\mathsf{G} : \{0,1\}^* \rightarrow \{0,1\}^{\mathsf{PKE.rl}}$ and $\overline{\mathsf{PKE}}.\mathsf{Kg} := \mathsf{PKE.Kg}$. **Right:** KEM scheme $\mathsf{KEM} = \mathbf{U}[\overline{\mathsf{PKE}}, s]$ obtained from the U-Transform, where KEM.ROS is the set of all (G, H) such that $\mathsf{G} \in \overline{\mathsf{PKE}}.\mathsf{ROS}$ and $\mathsf{H} : \{0,1\}^* \rightarrow \{0,1\}^s$. Further, $\mathsf{KEM.Kg} := \overline{\mathsf{PKE}}.\mathsf{Kg}$.

Due to its practical relevance, we want to extend this analysis to the muc setting, targeting a tightness loss in the number of corruptions. In order to do so, we make use of the modular approach of the FO transform as considered in [44]. The two transformations $\mathbf{T}$ and $\mathbf{U}$ are described in Fig. 8 and combining them gives us a KEM scheme $\mathsf{KEM} = \mathbf{U}[\mathbf{T}[\mathsf{PKE}], s]$. Our result is Theorem 11. Recall that PKE is γ-spread [26] if $\max_{C \in \{0,1\}^*} \Pr[\mathsf{PKE.Enc}(ek, M) = C] \leq 2^{-\gamma}$ for every $(ek, dk) \in \mathbf{Out}(\mathsf{PKE.Kg})$ and every $M \in \mathsf{PKE.MS}$, where the probability is taken over the coins of PKE.Enc.

Theorem 11. *Let* PKE *be* γ*-spread and* $\mathsf{KEM} = \mathbf{U}[\mathbf{T}[\mathsf{PKE}], s]$ *the KEM scheme obtained from applying the modular FO transform as described in Fig. 8. Let* n, c *be integers such that* $0 \leq c < n$*, and let* $m = \lfloor (n-1)/(c+1) \rfloor$*. Let* $\mathcal{A}$ *be an adversary for game* $\mathbf{G}_{\mathsf{KEM}}^{\mathsf{CCA\text{-}SB}\text{-muc-}(n,c)}$ *that issues at most* q_e *queries to* $\textsc{Oracle}(\mathtt{enc}, \cdot)$*,* q_d *queries to* $\textsc{Oracle}(\mathtt{dec}, \cdot)$ *and* q_{ro} *queries to both random oracles. Then we construct an adversary* $\mathcal{B}$ *for game* $\mathbf{G}_{\mathsf{PKE}}^{\mathsf{CPA\text{-}SB}\text{-mu-}m}$ *such that*

$$\mathbf{Adv}_{\mathsf{KEM}}^{\mathsf{CCA\text{-}SB}\text{-muc-}(n,c)}(\mathcal{A}) \leq 2e(c+1) \cdot \mathbf{Adv}_{\mathsf{PKE}}^{\mathsf{CPA\text{-}SB}\text{-mu-}m}(\mathcal{B}) + q_d \cdot 2^{-\gamma} + \frac{2nq_e q_{ro} + 1}{|\mathsf{PKE.MS}|} .$$

Adversary $\mathcal{B}$ *makes the same number of encryption queries as* $\mathcal{A}$ *makes. The running time of* $\mathcal{B}$ *is about that of* $\mathcal{A}$ *plus the time for maintaining lists of encryption, decryption and random oracle queries and the time for an execution of the sampler* $\mathsf{D\text{-}FX}$.

We provide the full proof in the full version [7]. Starting with a $\mathsf{CPA\text{-}SB}$-mu secure scheme PKE, step (1) applies transform $\mathbf{T}$ to obtain a $\mathsf{OW\text{-}PCVA}$-mu secure scheme $\overline{\mathsf{PKE}}$. In step (2), we apply Theorem 8 for $\mathsf{OW\text{-}PCVA}$. Finally, in step (3), we apply the second part of the transform, namely, we take the $\mathsf{OW\text{-}PCVA}$-muc secure scheme and use transform $\mathbf{U}$ to obtain the $\mathsf{CCA\text{-}SB}$-muc scheme KEM. We want to note that Jaeger [46] analyzes transform $\mathbf{U}$ in a simulation-based setting with corruptions, focusing on step (3) only.

Corruption-parameterized security for AKE. Security models for authenticated key exchange (AKE) protocols (e. g., [8]) allow for corruptions of long-term secret keys which makes the construction of tightly-secure AKE [2,32,36,47,56,64] non-trivial. All these works require some kind of muc security for their building blocks. As for signatures and encryption, we propose to study AKE with the number of corruptions being an additional adversary resource. This allows us to give improved tightness results for many AKE protocols.

In [16] Cohn-Gordon et al. analyze very efficient Diffie-Hellman based AKE protocols. We denote their main protocol by CCGJJ which is proven secure based on $\mathsf{St\text{-}CDH}$ with a loss linear in the number of users. They also show that this loss is optimal for their and similar DH-based protocols. By considering cp security, however, we can improve the previous bound to the number of corruptions. For this we make use of the analysis in [51] which proves tight security of the protocol based on the variant of $\mathsf{St\text{-}CDH}$ which we used for HEG. The improved bound for CCGJJ follows from combining our general theorem with [51, Thm. 3].

A common approach to generically construct AKE protocols is to use signatures for authentication, as for example in TLS. This means that long-term keys are now signing keys of a signature scheme. When aiming for tight security proofs, we thus require UF-muc security. We want to use our direct results for signatures here. Combining it with the results in [19,23,32,36], we get AKE with forward secrecy and with a security loss linear in the number of corruptions, requiring a UF-mu secure signature scheme. The other assumptions made remain untouched and their bound is already tight.

Improved bounds for selective opening security. In selective opening security, the adversary is given multiple PKE ciphertexts and is allowed to reveal not only the encrypted messages, but also the encryption randomness. We then want non-revealed ciphertexts to remain secure. In this setting, cp security concerns the number of opened ciphertexts. The security notion we are interested in is simulation-based selective opening security (in the following denoted by $\mathsf{SIM\text{-}SO\text{-}CCA}$) which is considered the strongest notion of security [5,14]. While our framework is designed for game-based rather than simulation-based security, we show how to leverage our framework to improve the bounds of the schemes considered in [42]. In particular, we show that their Diffie-Hellman based PKE scheme can be proven secure assuming $\mathsf{St\text{-}CDH}$ with a loss linear in the number of opened ciphertexts rather than the total number of ciphertexts. The scheme is conceptually simpler than the scheme of [65] (which has full tightness) and allows to save one group element in the ciphertext. Further, we show that the improved bound also applies to the RSA-based variant of the scheme. We provide formal definitions and theorem statements in the full version [7].

6 Optimality Results

We want to show that the loss of c is indeed optimal for a large class of schemes and games. For this, we use the framework of Bader, Jager, Li and Schäge [3]. We adapt their definitions to our framework of formal security specifications.

Re-randomizable relations. A relation Rel specifies a generation algorithm $\mathsf{Rel.Gen}$ that via $(x, \omega) \leftarrow\!\!\$\ \mathsf{Rel.Gen}$ outputs an instance x and witness ω. The (possibly randomized) verification algorithm $d \leftarrow\!\!\$\ \mathsf{Rel.Vf}(x, \omega')$ returns a boolean decision d as to whether ω' is a valid witness for x. Correctness asks that $\Pr[\mathsf{Rel.Vf}(\mathsf{Rel.Gen})] = 1$, where the probability is over the coins of $\mathsf{Rel.Gen}$ (and possibly $\mathsf{Rel.Vf}$). Let $\mathsf{Rel.WS}(x) = \{\, \omega : \mathsf{Rel.Vf}(x, \omega) = \mathsf{true} \,\}$ be the witness set of x. We say an algorithm $\mathsf{Rel.ReRand}$ is a *re-randomizer* for Rel if on input an instance x and a witness ω such that $\mathsf{Rel.Vf}(\omega, x) = \mathsf{true}$, it outputs a witness ω' which is uniformly distributed over $\mathsf{Rel.WS}(x)$ with probability 1.

We define a witness recovery game for a scheme Rel via FFS REC. Its formal description and muc game $\mathbf{G}^{\mathsf{REC}\text{-muc-}(n,c)}_{\mathsf{Rel}}$ are given in the full version [7]. The adversary in this game gets n statements and may ask for witnesses of c of them. Its task is then to compute a witness for an "uncorrupted" statement.

Simple adversaries and (black-box) reductions. Our optimality results will involve a special class of "simple" adversaries issuing all corruptions at once. More precisely, a (n, c)-*simple adversary* for relation Rel is a randomized and stateful algorithm sA, which induces an adversary $\mathcal{A} = \mathcal{A}[\mathsf{sA}]$ for $\mathbf{G}^{\mathsf{REC}\text{-muc-}(n,c)}_{\mathsf{Rel}}$ as described on the left-hand side of Fig. 9.

We also consider a class of natural black-box reductions that transform an (n, c)-simple adversary sA for Rel into an adversary for a game $\mathbf{G}$. Hence, in the following, $\mathbf{G}^{\mathsf{REC}\text{-muc-}(n,c)}_{\mathsf{Rel}}$ will play the role of the security game of the considered scheme, whereas $\mathbf{G}$ will be associated with some underlying assumption. More formally, an (n, c, r)-*simple* $(\mathbf{G} \to \mathbf{G}_{\mathsf{Rel}}^{\mathsf{REC}\text{-muc-}(n,c)})$-*reduction* sR is a stateful and randomized algorithm which, for every (n, c)-simple adversary sA, induces the adversary $\mathcal{R} = \mathcal{R}[\mathsf{sR}, \mathsf{sA}]$ described on the right of Fig. 9. Crucially, we require $\mathcal{R}$ to be a valid adversary for game $\mathbf{G}$. In particular, sR can make calls to the oracles provided by $\mathbf{G}$, except for calling its FIN procedure. Here, we do not allow the reduction to choose the random tape of the adversary, but note that it is easy to incorporate this into our formalization (and to extend Theorem 12 to take this into account) via standard techniques.[1]

```
Adversary $\mathcal{A}[\mathsf{sA}]$:
$(x_1, \ldots, x_n) \leftarrow\$ \mathbf{G}_{\mathsf{Rel}}^{\mathsf{REC}\text{-muc-}(n,c)}.\textsc{Init}$
$st \leftarrow \varepsilon$
$(\mathrm{CS}, st) \leftarrow\$ \mathsf{sA}(x_1, \ldots, x_n, st)$
If $|\mathrm{CS}| > c$ then abort
For all $i \in \mathrm{CS}$ do
    $\omega_i \leftarrow \mathbf{G}_{\mathsf{Rel}}^{\mathsf{REC}\text{-muc-}(n,c)}.\textsc{Corrupt}(i)$
$(i^*, \omega^*) \leftarrow\$ \mathsf{sA}((\omega_i)_{i \in \mathrm{CS}}, st)$
$\mathsf{win} \leftarrow \mathbf{G}_{\mathsf{Rel}}^{\mathsf{REC}\text{-muc-}(n,c)}.\textsc{Fin}(i^*, \omega^*)$
```

```
Adversary $\mathcal{R}[\mathsf{sR}, \mathsf{sA}]$:
$st_{\mathsf{sR}} \leftarrow \varepsilon$
For $k = 1, \ldots, r$ do
    $st_{\mathsf{sA}} \leftarrow \varepsilon$
    $(x_1, \ldots, x_n, \rho, st_{\mathsf{sR}}) \leftarrow\$ \mathsf{sR}(st_{\mathsf{sR}})$
    $(\mathrm{CS}, st_{\mathsf{sA}}) \leftarrow \mathsf{sA}(x_1, \ldots, x_n, st_{\mathsf{sA}})$
    $((\omega_i)_{i \in \mathrm{CS}}, st_{\mathsf{sR}}) \leftarrow\$ \mathsf{sR}(\mathrm{CS}, st_{\mathsf{sR}})$
    $(i^*, \omega^*) \leftarrow \mathsf{sA}((\omega_i)_{i \in \mathrm{CS}}, st_{\mathsf{sA}})$
    $st_{\mathsf{sR}} \leftarrow\$ \mathsf{sR}(i^*, \omega^*, st_{\mathsf{sR}})$
$\mathsf{win} \leftarrow\$ \mathbf{G}.\textsc{Fin}(st_{\mathsf{sR}})$
```

Fig. 9. Left: Adversary $\mathcal{A}[\mathsf{sA}]$ induced by an (n, c)-simple adversary sA for Rel. **Right:** Adversary $\mathcal{R}[\mathsf{sR}, \mathsf{sA}]$ defined by an (n, c, r)-simple reduction sR.

STATELESS GAMES. We prove our first tightness result, showing that any (n, c, r)-*simple* $(\mathbf{G} \to \mathbf{G}_{\mathsf{Rel}}^{\mathsf{REC}\text{-muc-}(n,c)})$-*reduction* sR must incur an advantage loss of roughly a factor $c+1$ for any re-randomizable relation Rel whenever the game $\mathbf{G}$ is *stateless*. A *stateless* game is one for which the INIT procedure sets some global variables, and queries to all game oracles never update these global variables, and the query output only depends on the global variables and the query input.

Our result will not require the game itself to be efficiently implementable, and often a game can have both an inefficient stateless version, as well as an equivalent efficient stateful version. For example, the efficient realization of the game defining PRF security would proceed by lazy sampling a random function, whereas an equivalent stateless version would initially sample a whole random function table. This is an example of game for which our result applies. Other games, such as those modeling the unforgeability of signatures under chosen message attacks, are inherently stateful, as the validity of the final forgery depends on previously issued signing queries.

Here, we will consider search games $\mathbf{G}$ (rather than distinguishing games) as the starting point for our reductions, and thus we associate an advantage metric

[1] More specifically, sR in Fig. 9 outputs the random coins to be used as input to sA. In the proof, we then make the meta-adversary deterministic, and use an efficient and secure pseudorandom function, with a hard coded key, to efficiently simulate the random choices based on its input.

$\mathbf{Adv}_{\mathbf{G}}(\mathcal{A}) = \Pr[\mathbf{G}(\mathcal{A})]$. However, it is straightforward to extend our result to distinguishing games.

Tightness result. The proof of the following theorem generalizes that of Bader et al. [3] by extending it to stateless games and to a setting with c out of n corruptions.

Theorem 12. *Let n, c be integers such that $0 \leq c < n$. Let $\mathbf{G}$ be a stateless game and let Rel be a relation with re-randomizer ReRand. There exists an (n, c)-simple adversary sA such that $\mathbf{Adv}^{\mathsf{REC}\text{-muc-}(n,c)}_{\mathsf{Rel}}(\mathcal{A}[\mathsf{sA}]) = 1$, and such that for any (n, c, r)-simple $(\mathbf{G} \rightarrow \mathbf{G}^{\mathsf{REC}\text{-muc-}(n,c)}_{\mathsf{Rel}})$-reduction sR, there exists an adversary $\mathcal{B}$ for $\mathbf{G}$ such that*

$$\mathbf{Adv}_{\mathbf{G}}(\mathcal{B}) \geq \mathbf{Adv}_{\mathbf{G}}(\mathcal{R}) - \frac{r}{c+1}$$

for $\mathcal{R} = \mathcal{R}[\mathsf{sR}, \mathsf{sA}]$. The number of queries $\mathcal{B}$ makes to the oracles in $\mathbf{G}$ is at most c times the number of queries sR makes. Further, the running time of $\mathcal{B}$ is at most $r \cdot (c + 1)$ times the running time of sR plus running the relation's verification algorithm $rc \cdot (c + 1)$ times plus running the re-randomizer once.

We give the full proof in the full version [7]. We start by describing an inefficient hypothetical adversary for game $\mathbf{G}^{\mathsf{REC}\text{-muc-}(n,c)}_{\mathsf{Rel}}$ which selects an index i^* of one of the first $c+1$ statements, learns the remaining c witnesses and then computes a random witness for i^*. We then describe a "meta-adversary" $\mathcal{B}$ which will *efficiently* simulate the adversary $\mathcal{R} = \mathcal{R}[\mathsf{sR}, \mathsf{sA}]$, running sR for each $i \in [c + 1]$ and r times. The simulation is perfect up to a failure probability $r/(c+1)$ which comes from the event that sR does not output a valid witness when run with i^*. The important observation is that sR can make calls to $\mathbf{G}$'s procedures, but because the game is stateless, these calls do not affect the overall state.

Interpretation of the above result. To explain optimality, let us define tightness more formally. Let sA be an (n, c)-simple adversary for $\mathbf{G}$ running in time $t_{\mathcal{A}} = t_{\mathsf{sA}}$ with advantage $\varepsilon_{\mathcal{A}} = \mathbf{Adv}^{\mathsf{REC}\text{-muc-}(n,c)}_{\mathsf{Rel}}(\mathcal{A}[\mathsf{sA}])$. Following [3], we say that an (n, c, r)-simple $(\mathbf{G} \rightarrow \mathbf{G}^{\mathsf{REC}\text{-muc-}(n,c)}_{\mathsf{Rel}})$-reduction sR running in time t_{sR} with advantage $\varepsilon_{\mathcal{R}} = \mathbf{Adv}_{\mathbf{G}}(\mathcal{R}[\mathsf{sR}, \mathsf{sA}])$ *loses* a factor ℓ if $t_{\mathcal{R}}/\varepsilon_{\mathcal{R}} \geq \ell \cdot t_{\mathcal{A}}/\varepsilon_{\mathcal{A}}$, where $t_{\mathcal{R}} = t_{\mathsf{sR}} + r \cdot t_{\mathsf{sA}}$.

Now we want to look closer at Theorem 12. Applying the bound gives us

$$\frac{t_{\mathcal{R}}}{\varepsilon_{\mathcal{R}}} = \frac{t_{\mathsf{sR}} + r \cdot t_{\mathsf{sA}}}{\varepsilon_{\mathcal{R}}} \geq \frac{r \cdot t_{\mathsf{sA}}}{\varepsilon_{\mathcal{B}} + r/(c+1)} = r\left(\varepsilon_{\mathcal{B}} + \frac{r}{c+1}\right)^{-1} \cdot \frac{t_{\mathsf{sA}}}{1} = \frac{r(c+1)}{\varepsilon_{\mathcal{B}}(c+1) + r} \cdot \frac{t_{\mathcal{A}}}{\varepsilon_{\mathcal{A}}} .$$

In the last step we use that the reduction must work for any (n, c)-simple adversary and, in particular, for the adversary sA with advantage 1 that we construct in the proof. We conclude that if $\varepsilon_{\mathcal{B}}$ is small, then $\ell \approx c + 1$ and $\mathcal{R}$ must lose a factor $c + 1$.

Example. The above theorem naturally covers non-interactive complexity assumptions as considered by [3] which are, by definition, stateless. As a more interesting example, the result also applies when using $\mathbf{G}_{\mathsf{KEM}}^{\mathsf{CPA\text{-}SB}\text{-mu-}n}$ for KEM schemes as a starting game $\mathbf{G}$ and a suitable relation, $\mathsf{Rel}[\mathsf{KEM}]$, which we provide in the full version [7]. That is, if there exists an efficient re-randomizer $\mathsf{Rel}[\mathsf{KEM}].\mathsf{ReRand}$, then any (n, c, r)-simple $(\mathbf{G}_{\mathsf{KEM}}^{\mathsf{CPA\text{-}SB}\text{-mu-}n} \rightarrow \mathbf{G}_{\mathsf{Rel[KEM]}}^{\mathsf{REC}\text{-muc-}(n,c)})$-reduction must lose a factor $c + 1$.

All schemes for which the above relation is unique, i. e., there exists exactly one decryption key such that the relation holds, have a trivial efficient re-randomizer. An example is the ElGamal KEM.

In order to conclude that the optimality result also holds when the target game is the corresponding muc game (rather than the relation game), we describe in the full version [7] how to transform *any* simple adversary sA for $\mathbf{G}_{\mathsf{Rel[KEM]}}^{\mathsf{REC}\text{-muc-}(n,c)}$ into an equivalent adversary $\mathcal{A}'[\mathsf{sA}]$ for game $\mathbf{G}_{\mathsf{KEM}}^{\mathsf{CPA\text{-}SB}\text{-muc-}(n,c)}$. Following [3], the existence of a black-box reduction from $\mathbf{G}_{\mathsf{KEM}}^{\mathsf{CPA\text{-}SB}\text{-mu-}n}$ to $\mathbf{G}_{\mathsf{KEM}}^{\mathsf{CPA\text{-}SB}\text{-muc-}(n,c)}$, rewinding the given adversaries r times, implies the existence of an (n, c, r)-simple $(\mathbf{G}_{\mathsf{KEM}}^{\mathsf{CPA\text{-}SB}\text{-mu-}n} \rightarrow \mathbf{G}_{\mathsf{Rel[KEM]}}^{\mathsf{REC}\text{-muc-}(n,c)})$-reduction sR with the same loss. Roughly, this is because we can build sR which, when given access to sA, behaves exactly the same as the reduction using $\mathcal{A}'[\mathsf{sA}]$. As in [3], we do not make this argument fully formal, as it requires formalizing a more general notion of black-box reduction.

GENERALIZING BEYOND STATELESS GAMES. Since many games of interest are not stateless, we aim to further generalize Theorem 12. For this, we introduce the notion of a *branching adversary* for $\mathbf{G}$ which describes a pair of adversaries $(\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}})$. Adversary $\mathsf{B}_{\mathsf{main}}$ is stateful, and proceeds in stages. At each step, adversary $\mathsf{B}_{\mathsf{side}}$ is also executed, but its output is ignored. We are then interested in how much the actions of $\mathsf{B}_{\mathsf{side}}$ alter the execution of the adversary as defined by $\mathsf{B}_{\mathsf{main}}$, which we capture as an advantage function $\mathbf{Adv}_{\mathbf{G}}^{\mathrm{branch}}(\mathsf{B}_{\mathsf{main}}, \mathsf{B}_{\mathsf{side}})$. If this advantage is small for a suitable branching adversary, then an adapted version of Theorem 12 is still meaningful. We provide a formal treatment in the full version [7].

Using this generalization, we are able to show optimality results for strongly-unforgeable randomized signatures and IND-CCA secure KEMs, where we bound the branching advantage by the entropy of signatures resp. ciphertexts.

Acknowledgments. Bellare was supported in part by NSF grant CNS-2154272 and KACST. Work done while Riepel was at UCSD, supported in part by KACST. Tessaro was supported in part by NSF grants CNS-2026774, CNS-2154174, a JP Morgan Faculty Award, a CISCO Faculty Award, and a gift from Microsoft.

References

1. Abdalla, M., Bellare, M., Rogaway, P.: The oracle Diffie-Hellman assumptions and an analysis of DHIES. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 143–158. Springer, Heidelberg (Apr 2001). https://doi.org/10.1007/3-540-45353-9_12

2. Bader, C., Hofheinz, D., Jager, T., Kiltz, E., Li, Y.: Tightly-secure authenticated key exchange. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 629–658. Springer, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-46494-6_26
3. Bader, C., Jager, T., Li, Y., Schäge, S.: On the impossibility of tight cryptographic reductions. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 273–304. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_10
4. Bellare, M., Boldyreva, A., Micali, S.: Public-key encryption in a multi-user setting: Security proofs and improvements. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 259–274. Springer, Heidelberg (May 2000). https://doi.org/10.1007/3-540-45539-6_18
5. Bellare, M., Dowsley, R., Waters, B., Yilek, S.: Standard security does not imply security against selective-opening. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 645–662. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_38
6. Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. In: Arge, L., Cachin, C., Jurdzinski, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 411–422. Springer, Heidelberg (Jul 2007). https://doi.org/10.1007/978-3-540-73420-8_37
7. Bellare, M., Riepel, D., Tessaro, S., Zhang, Y.: Count corruptions, not users: Improved tightness for signatures, encryption and authenticated key exchange. Cryptology ePrint Archive, Paper 2024/1258 (2024), https://eprint.iacr.org/2024/1258
8. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D.R. (ed.) CRYPTO'93. LNCS, vol. 773, pp. 232–249. Springer, Heidelberg (Aug 1994). https://doi.org/10.1007/3-540-48329-2_21
9. Bellare, M., Rogaway, P.: The exact security of digital signatures: How to sign with RSA and Rabin. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 399–416. Springer, Heidelberg (May 1996). https://doi.org/10.1007/3-540-68339-9_34
10. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_25
11. Bernstein, D.J.: Multi-user Schnorr security, revisited. Cryptology ePrint Archive, Report 2015/996 (2015), https://eprint.iacr.org/2015/996
12. Bernstein, D.J., Duif, N., Lange, T., Schwabe, P., Yang, B.Y.: High-speed high-security signatures. Journal of cryptographic engineering **2**(2), 77–89 (2012)
13. Blakley, G.R.: Safeguarding cryptographic keys. Proceedings of AFIPS 1979 National Computer Conference **48**, 313–317 (1979)
14. Böhl, F., Hofheinz, D., Kraschewski, D.: On definitions of selective opening security. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012. LNCS, vol. 7293, pp. 522–539. Springer, Heidelberg (May 2012). https://doi.org/10.1007/978-3-642-30057-8_31
15. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehle, D.: CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367 (2018)

16. Cohn-Gordon, K., Cremers, C., Gjøsteen, K., Jacobsen, H., Jager, T.: Highly efficient key exchange protocols with optimal tightness. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 767–797. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26954-8_25
17. Coron, J.S.: On the exact security of full domain hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (Aug 2000). https://doi.org/10.1007/3-540-44598-6_14
18. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing **33**(1), 167–226 (2003)
19. Davis, H., Günther, F.: Tighter proofs for the SIGMA and TLS 1.3 key exchange protocols. In: Sako, K., Tippenhauer, N.O. (eds.) ACNS 21, Part II. LNCS, vol. 12727, pp. 448–479. Springer, Heidelberg (Jun 2021). https://doi.org/10.1007/978-3-030-78375-4_18
20. De Santis, A., Desmedt, Y., Frankel, Y., Yung, M.: How to share a function securely. In: 26th ACM STOC. pp. 522–533. ACM Press (May 1994). https://doi.org/10.1145/195058.195405
21. Dent, A.W.: A designer's guide to KEMs. In: Paterson, K.G. (ed.) 9th IMA International Conference on Cryptography and Coding. LNCS, vol. 2898, pp. 133–151. Springer, Heidelberg (Dec 2003)
22. Diemert, D., Gellert, K., Jager, T., Lyu, L.: More efficient digital signatures with tight multi-user security. In: Garay, J. (ed.) PKC 2021, Part II. LNCS, vol. 12711, pp. 1–31. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75248-4_1
23. Diemert, D., Jager, T.: On the tight security of TLS 1.3: Theoretically sound cryptographic parameters for real-world deployments. Journal of Cryptology **34**(3), 30 (Jul 2021). https://doi.org/10.1007/s00145-021-09388-x
24. Dragon, S.: Top 12 revealing ssl stats you should know (May 2023), https://www.ssldragon.com/blog/ssl-stats/
25. Duman, J., Hövelmanns, K., Kiltz, E., Lyubashevsky, V., Seiler, G.: Faster lattice-based KEMs via a generic fujisaki-okamoto transform using prefix hashing. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 2722–2737. ACM Press (Nov 2021). https://doi.org/10.1145/3460120.3484819
26. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M.J. (ed.) CRYPTO'99. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (Aug 1999). https://doi.org/10.1007/3-540-48405-1_34
27. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. Journal of Cryptology **26**(1), 80–101 (2013). https://doi.org/10.1007/s00145-011-9114-1
28. Gay, R., Hofheinz, D., Kiltz, E., Wee, H.: Tightly CCA-secure encryption without pairings. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part I. LNCS, vol. 9665, pp. 1–27. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49890-3_1
29. Gay, R., Hofheinz, D., Kohl, L.: Kurosawa-desmedt meets tight security. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part III. LNCS, vol. 10403, pp. 133–160. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63697-9_5
30. Gellert, K., Gjøsteen, K., Jacobsen, H., Jager, T.: On optimal tightness for key exchange with full forward secrecy via key confirmation. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 297–329. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38551-3_10

31. Giacon, F., Kiltz, E., Poettering, B.: Hybrid encryption in a multi-user setting, revisited. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part I. LNCS, vol. 10769, pp. 159–189. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76578-5_6
32. Gjøsteen, K., Jager, T.: Practical and tightly-secure digital signatures and authenticated key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 95–125. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_4
33. Goh, E.J., Jarecki, S., Katz, J., Wang, N.: Efficient signature schemes with tight reductions to the Diffie-Hellman problems. Journal of Cryptology **20**(4), 493–514 (Oct 2007). https://doi.org/10.1007/s00145-007-0549-3
34. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: Aho, A. (ed.) 19th ACM STOC. pp. 218–229. ACM Press (May 1987). https://doi.org/10.1145/28395.28420
35. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing **17**(2), 281–308 (Apr 1988)
36. Han, S., Jager, T., Kiltz, E., Liu, S., Pan, J., Riepel, D., Schäge, S.: Authenticated key exchange and signatures with tight security in the standard model. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 670–700. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84259-8_23
37. Han, S., Liu, S., Gu, D.: Key encapsulation mechanism with tight enhanced security in the multi-user setting: Impossibility result and optimal tightness. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part II. LNCS, vol. 13091, pp. 483–513. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92075-3_17
38. Han, S., Liu, S., Gu, D.: Almost tight multi-user security under adaptive corruptions & leakages in the standard model. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 132–162. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30620-4_5
39. Han, S., Liu, S., Lyu, L., Gu, D.: Tight leakage-resilient CCA-security from quasi-adaptive hash proof system. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 417–447. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_15
40. Han, S., Liu, S., Wang, Z., Gu, D.: Almost tight multi-user security under adaptive corruptions from LWE in the standard model. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 682–715. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_22
41. Hanaoka, G., Schuldt, J.C.N.: On signatures with tight security in the multi-user setting. In: 2016 International Symposium on Information Theory and Its Applications (ISITA). pp. 91–95 (2016)
42. Heuer, F., Jager, T., Kiltz, E., Schäge, S.: On the selective opening security of practical public-key encryption schemes. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 27–51. Springer, Heidelberg (Mar / Apr 2015). https://doi.org/10.1007/978-3-662-46447-2_2
43. Heum, H., Stam, M.: Tightness subtleties for multi-user PKE notions. In: Paterson, M.B. (ed.) 18th IMA International Conference on Cryptography and Coding. LNCS, vol. 13129, pp. 75–104. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92641-0_5

44. Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the Fujisaki-Okamoto transformation. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 341–371. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_12
45. Hofheinz, D., Jager, T.: Tightly secure signatures and public-key encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 590–607. Springer, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_35
46. Jaeger, J.: Let attackers program ideal models: Modularity and composability for adaptive compromise. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 101–131. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30620-4_4
47. Jager, T., Kiltz, E., Riepel, D., Schäge, S.: Tightly-secure authenticated key exchange, revisited. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 117–146. Springer, Heidelberg (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_5
48. Jager, T., Stam, M., Stanley-Oakes, R., Warinschi, B.: Multi-key authenticated encryption with corruptions: Reductions are lossy. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017, Part I. LNCS, vol. 10677, pp. 409–441. Springer, Heidelberg (Nov 2017). https://doi.org/10.1007/978-3-319-70500-2_14
49. Josefsson, S., Liusvaara, I.: Edwards-curve digital signature algorithm (EdDSA). RFC 8032 (Jan 2017), https://datatracker.ietf.org/doc/html/rfc8032
50. Kiltz, E., Masny, D., Pan, J.: Optimal security proofs for signatures from identification schemes. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part II. LNCS, vol. 9815, pp. 33–61. Springer, Heidelberg (Aug 2016). https://doi.org/10.1007/978-3-662-53008-5_2
51. Kiltz, E., Pan, J., Riepel, D., Ringerud, M.: Multi-user CDH problems and the concrete security of NAXOS and HMQV. In: Rosulek, M. (ed.) CT-RSA 2023. LNCS, vol. 13871, pp. 645–671. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30872-7_25
52. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_33
53. Lacharité, M.S.: Security of BLS and BGLS signatures in a multi-user setting. Cryptography and Communications **10**(1), 41–58 (2018). https://doi.org/10.1007/s12095-017-0253-6
54. Lee, Y., Lee, D.H., Park, J.H.: Tightly CCA-secure encryption scheme in a multi-user setting with corruptions. DCC **88**(11), 2433–2452 (2020). https://doi.org/10.1007/s10623-020-00794-z
55. Libert, B., Joye, M., Yung, M., Peters, T.: Concise multi-challenge CCA-secure encryption and signatures with almost tight security. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part II. LNCS, vol. 8874, pp. 1–21. Springer, Heidelberg (Dec 2014). https://doi.org/10.1007/978-3-662-45608-8_1
56. Liu, X., Liu, S., Gu, D., Weng, J.: Two-pass authenticated key exchange with explicit authentication and tight security. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 785–814. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_27
57. Menezes, A., Smart, N.: Security of signature schemes in a multi-user setting. Designs, Codes and Cryptography **33**(3), 261–274 (2004)
58. Microsoft: Results of major technical investigations for storm-0558 key acquisition. Microsoft Blog (September 2023), https://msrc.microsoft.com/blog/2023/09/results-of-major-technical-investigations-for-storm-0558-key-acquisition/

59. Morgan, A., Pass, R., Shi, E.: On the adaptive security of MACs and PRFs. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part I. LNCS, vol. 12491, pp. 724–753. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64837-4_24
60. National Institute for Standards and Technology (NIST): Post-quantum cryptography standardization, https://csrc.nist.gov/projects/post-quantum-cryptography
61. National Institute of Standards and Technology: Digital Signature Standard (DSS). FIPS PUB 186-5 (Feb 2023), https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-5.pdf
62. Pan, J., Ringerud, M.: Signatures with tight multi-user security from search assumptions. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) ESORICS 2020, Part II. LNCS, vol. 12309, pp. 485–504. Springer, Heidelberg (Sep 2020). https://doi.org/10.1007/978-3-030-59013-0_24
63. Pan, J., Wagner, B.: Lattice-based signatures with ti ght adaptive corruptions and more. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022, Part II. LNCS, vol. 13178, pp. 347–378. Springer, Heidelberg (Mar 2022). https://doi.org/10.1007/978-3-030-97131-1_12
64. Pan, J., Wagner, B., Zeng, R.: Lattice-based authenticated key exchange with tight security. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 616–647. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_20
65. Pan, J., Zeng, R.: Compact and tightly selective-opening secure public-key encryption schemes. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part III. LNCS, vol. 13793, pp. 363–393. Springer, Heidelberg (Dec 2022). https://doi.org/10.1007/978-3-031-22969-5_13
66. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of Cryptology **4**(3), 161–174 (Jan 1991). https://doi.org/10.1007/BF00196725
67. Shamir, A.: How to share a secret. Communications of the Association for Computing Machinery **22**(11), 612–613 (Nov 1979). https://doi.org/10.1145/359168.359176
68. Whittacker, Z.: Microsoft lost its keys, and the government got hacked. TechCrunch (July 2023), https://techcrunch.com/2023/07/17/microsoft-lost-keys-government-hacked/
69. Zaverucha, G.: Hybrid encryption in the multi-user setting. Cryptology ePrint Archive, Report 2012/159 (2012), https://eprint.iacr.org/2012/159

Interval Key-Encapsulation Mechanism

Alexander Bienstock[1(✉)], Yevgeniy Dodis[2], Paul Rösler[3], and Daniel Wichs[4]

[1] J.P. Morgan AI Research and J.P. Morgan AlgoCRYPT CoE, New York, USA
alex.bienstock@jpmchase.com

[2] New York University, New York, USA
dodis@cs.nyu.edu

[3] FAU Erlangen-Nürnberg, Erlangen, Germany
paul.roesler@fau.de

[4] Northeastern University and NTT Research, Sunnyvale, USA
wichs@ccs.neu.edu

Abstract. *Forward-Secure* Key-Encapsulation Mechanism (FS-KEM; Canetti et al. Eurocrypt 2003) allows Alice to encapsulate a key k to Bob for some time t such that Bob can decapsulate it at any time $t' \leq t$. Crucially, a corruption of Bob's secret key after time t does not reveal k.

In this work, we generalize and extend this idea by also taking *Post-Compromise Security* (PCS) into account and call it *Interval Key-Encapsulation Mechanism* (IKEM). Thus, we do not only protect confidentiality of previous keys against future corruptions but *also* confidentiality of future keys against past corruptions. For this, Bob can regularly renew his secret key and inform others about the corresponding public key. IKEM enables Bob to decapsulate keys sent to him over an interval of time extending into the past, in case senders have not obtained his latest public key; forward security only needs to hold with respect to keys encapsulated before this interval. This basic IKEM variant can be instantiated based on standard KEM, which we prove to be optimal in terms of assumptions as well as ciphertext and key sizes.

We also extend this notion of IKEM for settings in which Bob decapsulates (much) later than Alice encapsulates (e.g., in high-latency or segmented networks): if a third user Charlie forwards Alice's ciphertext to Bob and, additionally, knows a recently renewed public key of Bob's, Charlie could re-encrypt the ciphertext for better PCS. We call this extended notion IKEM*R*. Our first IKEMR construction based on trapdoor permutations has (almost) constant sized ciphertexts in the number of re-encryptions; and our second IKEMR construction based on FS-PKE has constant sized public keys in the interval size.

Finally, to bypass our lower bound on the IKEM(R) secret key size, which must be linear in the interval size, we develop a new *Interval RAM* primitive with which Bob only stores a constant sized part of his secret key locally, while outsourcing the rest to a (possibly adversarial) server.

The full version [8] of this article is available in the IACR eprint archive as article 2024/1454, at https://eprint.iacr.org/2024/1454.

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15485, pp. 361–393, 2025.
https://doi.org/10.1007/978-981-96-0888-1_12

For all our constructions, we achieve security against active adversaries. For this, we obtain new insights on Replayable CCA security for KEM-type primitives, which might be of independent interest.

1 Introduction

Corruption of user secrets is an acknowledged threat in the cryptographic literature. Especially cryptographic protocols for secure long-term communication, such as secure messaging, implement measures to mitigate the effect of temporary user corruptions. The traditional goal *Forward Security* (FS) requires that prior communication remains secure even if user secrets are corrupted in the future. Modern communication protocols additionally fulfill *Post-Compromise Security* (PCS): users recover from earlier corruptions such that future communication will be secure again. Intuitively, this means that a corruption only reveals a (short) *interval* of secrets. To achieve these goals, protocols usually combine two techniques: (1) evolving secrets with one-way functions and then deleting the old secrets for FS, (2) randomly sampling fresh secrets and sharing the corresponding public values for PCS.

Session-Based Interval Security. The most prominent instantiation of this approach for two-party communication is the *Double Ratchet Algorithm* [30], implemented in Signal, WhatsApp, and many other messaging apps. A generalization based on tree-hierarchies for group communication is the *Messaging Layer Security* (MLS) standard [6].

Remarkably, both protocols and all their variants (e.g., [2,3,7,26,27,31–33]) are session-based. This means that each new conversation is established independent of existing ones such that every user stores a separate collection of secrets for each session it participates in. Due to this, evolving and deleting old secrets as well as sampling fresh secrets is conducted separately per session, which induces a linear communication overhead in the number of sessions per user. For illustration, consider a user Bob who is concerned that all secrets locally stored on his device were corrupted. To recover from this corruption, Bob samples independent, fresh secrets and shares corresponding public values in all his sessions.

To reduce this overhead, Alwen et al. [1] investigate how to merge overlapping structures of session secrets in group communication protocols underlying MLS. Intuitively, they exploit that the tree-hierarchy in MLS can unify overlapping sets of members for multiple (independent) group sessions. Still, their saving is modest for most hierarchy structures, and does not generally result in sublinear storage in the number of sessions.

Interval Security with Single Key. In this work, we avoid any session separation and instead let each user have a consolidated secret key with a corresponding public key, which basically describes the concept of public-key encryption or, in our work, *Key-Encapsulation Mechanism* (KEM). Providing FS for KEM with a *static* public key—via FS-KEM—has already been solved by Canetti et al. [17]. To also achieve PCS, the public key cannot remain static but must be updated. Thus, we will allow the recipient Bob to periodically update his public key. While this introduces some additional complexity in fetching Bob's latest key before sending him a message, we will see that this relaxation has many benefits, even beyond achieving PCS: for example, it results in noticeably more efficient schemes, by considerably simplifying achieving the FS aspect compared to the static public key model of [17]. Additionally, our model will be sufficiently flexible to allow Bob to still decapsulate ciphertexts in many cases when the senders failed to obtain Bob's latest public key.

More concretely, our new model is the following. Bob starts by generating a key pair (sk_0, pk_0) and shares pk_0 with all users who want to talk to him. Whenever Bob thinks he was corrupted, he uses his current secret key sk_{i-1} to derive a new key pair (sk_i, pk_i) and shares the new pk_i again. Using sk_i, Bob can still decapsulate keys encapsulated with pk_j for $j \leq i$. However, if he thinks that all keys encapsulated with public keys $pk_j, j < i^*$ were decapsulated already or should not be decapsulatable any longer, Bob can shorten the secret key interval from $[0, i]$ to $[i^*, i]$. After this, Bob can only decapsulate ciphertexts encapsulated to a public key pk_j such that $i^* \leq j \leq i$. Conversely, security requires that a corruption of current secret key sk_i will not affect ciphertexts encapsulated to pk_j for $j < i^*$ or $j > i$. Since Bob can continuously renew his key pair to start new *epochs* and shorten the interval of decapsulatable old *epochs*, we call this primitive *Interval KEM* (IKEM).

KEM-Based IKEM. Our simple KEM-based IKEM construction almost naturally follows the above described abstract syntax: whenever Bob (re-)generates his IKEM key pair, he simply generates a fresh KEM key pair, shares the new KEM public key, and adds the new KEM secret key to his IKEM secret key. To shorten the decapsulation interval, Bob just removes old KEM secret keys from his IKEM secret key. Since this idea is straight forward, and to save space, the full details of this construction and a formal security analysis are in the full version [8].

Secret Key Lower Bound. Notice, the extremely simple KEM-based IKEM from above has ciphertexts and public keys of constant size. However, the secret key grows linearly in the size of the current interval. One may wonder if this dependence is inherent. Unfortunately, as our first result we give the affirmative answer to this question: any IKEM secret key must be proportional to the size of the decapsulation interval.

One way to show this lower bound would be to prove that IKEM implies the simpler symmetric-key primitive called *Self Encrypted Queue* from Choi et al. [19]. This primitive also involves a secret state that can be updated for PCS, however, it only allows the receiver to encrypt keys to *itself* for future use (hence why it is symmetric-key). Self Encrypted Queue also requires the newly updated states to be able to decrypt the keys encrypted to old states, and thus the state of any construction seemingly needs to grow proportionally to the number of epochs, just as with our IKEM construction. Indeed, [19] shows that this is inherent for Self Encrypted Queues, by proving a lower bound showing that states need to be of size $f \cdot \lambda$, where f is the number of epochs.

Instead, we choose to prove a direct lower bound to show that IKEM secret keys need to grow with the size of the current interval, f, as we believe it more directly elucidates why this is in fact the case. Indeed, we use an encoding argument in the full version [8] to accomplish this, with intuition as follows: The encoder's goal is to succinctly encode and send a random bit string s to the decoder, who then must obtain s. To this end, the encoder and decoder initially share public randomness (independent of s) consisting of an initial IKEM key pair, as well as two lists of f random bit strings each. To encode the ith bit of string s, the encoder selects the ith random bit string of one of the two lists—the first list *iff* $s_i = 0$. With each selected bit string, the encoder re-generates the current IKEM key pair. The final IKEM secret key sk^* is the code. The decoding algorithm also starts with the initial IKEM key pair. For every bit, it re-generates the current IKEM key pair twice: once with each next random bit string from the two lists. Using the two resulting public keys, it encapsulates individual keys and trial-decapsulates both resulting ciphertexts with the code sk^*. By correctness, decapsulation of the right ciphertext yields the matching encapsulated key. By security, decapsulation of the wrong ciphertext yields a random key (that is independent of the encapsulated key). This procedure is repeated with the right IKEM key pairs until string s is decoded entirely. The formal proof in the full version [8] shows that the secret key size is linear in $f \cdot \lambda$, where f is size of the current decapsulation interval and λ is the security parameter.

Strengthening IKEM Security. While the lower bound on the secret key size of IKEM is unfortunate, in many settings the receiver can afford the extra storage, as this storage is local, and does not result in any increased network latency. Moreover, we will shortly describe a method to reduce the secret key storage by outsourcing. Yet, first we focus on extending the security of IKEM, by considering several motivating application scenarios.

First, in settings with high-latency or a segmented network topology, recovery by publishing a new public key for Bob may still be too slow: the new key may reach Bob's session partners only with a considerable delay. As an example, consider a decentralized gossiping or mesh network in which client devices exchange and forward traffic only with some contacts or with direct neighbors in their physical range (e.g., via bluetooth protocols). Based on this, a ciphertext from Alice to Bob is transmitted via multiple devices that span a delivery route in the network between them. Bienstock et al. [11] observe that this topology allows contacts to cooperate with each other to strengthen the security of forwarded ciphertexts: if a user Charlie processes a ciphertext c from Alice to Bob after Charlie received the most recent public key for Bob, Charlie can re-encrypt c using the information contained in Bob's latest key. Thus, even if Alice sent c when Bob was corrupted, Charlie's re-encryption after Bob's key update protects c on the remaining delivery route from Charlie to Bob.

This form of contact cooperation can also strengthen security of encrypted cloud storage, where the cloud provider acts as Charlie. Every user Alice and Bob can upload encrypted data to Bob's online folder. Bob can regularly re-new his key material and share the latest public key with the provider. Without any further interaction, the provider can re-encrypt all files in Bob's folder such that they remain secure even if Bob's old secrets are ever corrupted. Note that this is the simplest, essential form of one of the motivating examples for Proxy Re-Encryption (PRE) [13].[1] We elaborate on the relation to PRE at the end of this section.

More generally, contact cooperation can strengthen any high-latency delivery or long-time storage during which ciphertexts are processed by intermediate honest parties. For example, Alice can encapsulate a ciphertext c_1 for Bob at time t_1, and leave it with an intermediary Charlie (e.g., a notary), instead of immediately delivering it to Bob. Charlie is then instructed to deliver c_1 to Bob some time t_2 in the future; e.g., if a certain condition is triggered. With traditional encapsulation, the key in c_1 would be compromised if Bob is corrupted any time between t_1 and t_2. With IKEM syntax, however, it might be possible for Charlie to update c_1 into a "more secure variant" c'_1, using Bob's latest public key at epoch $t_2 > t_1$, and then only keep c'_1 until it is released to Bob.

Ciphertext Re-Encapsulation. Motivated by this form of contact cooperation, we add re-encapsulation to IKEM and call it IKEMR. A ciphertext c_1 encapsulated to Bob at time t_1 and re-encapsulated in ciphertext c'_1 at time $t_2 > t_1$ remains secure even if Bob was corrupted at any time $t^* < t_2$ as long as the adversary only ever sees c'_1. Furthermore, when Bob shortens the decapsulation interval of his secret key to exclude time t_1 for FS, future corruptions of Bob will not affect c'_1 either. Thereby, re-encapsulations do not extend the lifetime of a ciphertext: if the epoch t_1 at which a ciphertext was *originally* created falls out of the decapsulation interval, it cannot be decapsulated anymore even if it was re-encapsulated at later epochs $t_2 > t_1$. We illustrate this security requirement with a simple example in Fig. 1. Finally, a ciphertext can be re-encapsulated multiple times such that the first ciphertext version observed by the adversary determines the window of harmful and harmless corruptions, respectively. While this form of re-encapsulation ostensibly is closely related to Proxy Re-Encryption (PRE), we discuss the crucial differences at the end of this section.

Simple Extension Fails. Naively, one could add re-encapsulation to our KEM-based IKEM construction from above as follows: to re-encapsulate ciphertext c_1 from time t_1 with the most recent public key pk_2 from time $t_2 > t_1$, one simply uses the KEM to encapsulate a key k' in ciphertext c'_{KM} and then encrypts c_1 symmetrically with key k'. Thus, $c'_1 = c'_{\mathrm{KM}} \| \mathrm{E}_{k'}(c_1)$ is the re-encapsulated ciphertext, where $(k', c'_{\mathrm{KM}}) = \mathrm{KM.enc}(pk_2)$. To shorten the decapsulation interval, Bob still just removes the KEM secret keys of all abandoned epochs from his IKEMR secret key.

Now consider the case that Alice creates two ciphertexts c_1 and c_2 to Bob, one at time t_1 with pk_1 and one at time $t_2 > t_1$ with pk_2. Additionally, Charlie re-encapsulates c_1 at time t_2 in ciphertext c'_1. When Bob shortens his

[1] It is also a naturally useful variant of Updatable Encryption [15] for the public-key setting.

decapsulation interval from $[0, t_2]$ to $[t_2, t_2]$, correctness requires that c_2 can still be decapsulated. Furthermore, security requires that k_1 in c_1' remains confidential if the adversary only sees c_1' (but not c_1), even if Bob is corrupted before time t_2 and/or after he shortened his decapsulation interval to $[t_2, t_2]$ (the reader can again refer to Fig. 1 for an illustration of this security requirement). Yet, a corruption at time $t_1 < t_2$ for the above KEM-based IKEMR construction reveals KEM secret key sk_1; another corruption after the decapsulation interval was shortened to $[t_2, t_2]$ reveals KEM secret key sk_2. Thus, the adversary can remove the re-encapsulation layer of c_1' using sk_2 and then decapsulate the original ciphertext c_1 using sk_1. Hence, this simple extension of our KEM-based IKEM construction is insecure.

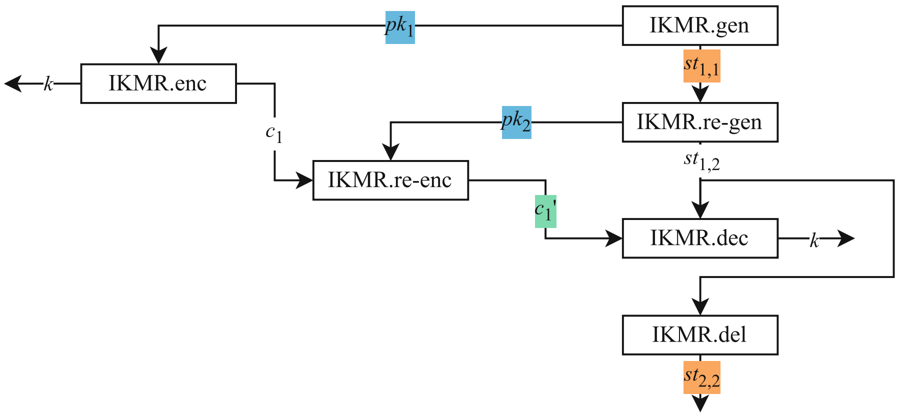


Fig. 1. Execution example for IKEMR: if the adversary only sees pk_1, pk_2, and c_1', and corrupts $st_{1,1}$, and $st_{2,2}$, key k is required to remain secure.

KEM-BASED IKEMR. The core problem of the above IKEMR construction is that it processes encapsulations and *re*-encapsulations at time t_2 identically. However, FS requires that Bob can shorten his decapsulation interval such that only c_2, originally created at time t_2, can be decapsulated but not the re-encapsulation c_1' from time t_2 of earlier ciphertext c_1, originally from time t_1.

To solve this problem, Bob, instead, generates multiple *extra* KEM key pairs at every IKEMR key re-generation: one KEM key pair for each epoch in his current decapsulation interval of length f. The resulting re-generated IKEMR public key $pk_t = (pk_{t,1}, \ldots, pk_{t,f})$ for time t consists of f fresh KEM public keys. (Re-)Encapsulation with IKEMR public key pk_t at time t then depends on the initial encapsulation time of a ciphertext: for ciphertext c initially created at time $t^* \leq t$ ($t^* = t$ for an initial encapsulation), c is (re-)encapsulated using KEM public key $pk_{t,t-t^*+1}$. Whenever Bob shortens his decapsulation interval to $[t_1, t_2]$, he first removes all secret keys generated before time t_1. Then, for each epoch t' with $t_1 \leq t' \leq t_2$, he removes every secret key $sk_{t',\tau}$ for which

$\tau > t' - t_1 + 1$. Thus, intuitively, every epoch's IKEMR public key contains extra KEM public keys that are exclusively used for *re*-encapsulations of ciphertexts initially created in prior epochs. As soon as such prior epochs are removed from the decapsulation interval, all corresponding extra KEM secret keys are removed from the IKEMR secret key.

By deriving the extra KEM secret keys at each IKEMR re-generation iteratively via a chain of Pseudo-Randomness Generators (PRGs), Bob only ever needs to store one KEM secret key and one PRG seed for each epoch in his current decapsulation interval. To shorten the decapsulation interval, Bob moves the PRG-chain of each epoch in the interval forward, thereby derives new KEM secret keys and deletes past KEM secret keys. Thus, the IKEMR secret key size is optimal based on our lower bound.

IKEMR with Small Public Keys. The advantage of this secure IKEMR construction is its mild underlying standard assumption: an ordinary KEM. Nevertheless, the disadvantages are: ciphertexts grow linearly in the number of re-encapsulations and public keys grow linearly in the decapsulation interval length. Using FS-PKE instead of ordinary KEM, we can reduce the size of IKEMR public keys. The intuition is that all *extra* KEM key pairs generated for one IKEMR key re-generation are consolidated in a single FS-PKE key pair. This shrinks the IKEMR public key to constant size but slightly increases the secret key to size $f \log f$ (instead of f), where f is the current length of the decapsulation interval. At (re-)encapsulation, instead of choosing the right *extra* KEM public key from the current IKEMR public key, Alice indicates the ciphertext's initial creation time when encrypting to the corresponding epoch of the single FS-PKE public key. When shortening the decapsulation interval, Bob simply updates all FS-PKE secret keys that remain in his IKEMR secret key such that epochs outside the shortened interval cannot be (re-)decapsulated. The full details of this construction are in Sect. 5.

IKEMR with Small Ciphertexts. Using Trapdoor Permutations (TDPs), we build an IKEMR construction with (almost) constant sized ciphertexts. For this, we begin with our KEM-based IKEMR construction from above that has linear sized public keys in the decapsulation interval length. Then, instead of using KEMs, we employ a family of TDPs with a common domain, where each KEM public key is replaced with the public key of a TDP from that family and each KEM secret key is replaced with the corresponding trapdoor. To encapsulate a key, this construction samples a random element from the TDP family's domain and evaluates the current epoch's TDP on it, which yields the ciphertext. The actual encapsulated key is derived by applying a randomness extractor to the random input element. To re-encapsulate a ciphertext, another TDP from the family is applied to this ciphertext. Since all permutations from the TDP family share the same domain, the (cryptographic part of the) ciphertext has constant size. Yet, for decapsulation, the receiver Bob needs to know which trapdoors he should use. Thus, at each re-encapsulation, the index of the current epoch is attached to the ciphertext, which increases the ciphertext linearly in the number of re-encapsulations—however, note that each attached index has only logarithmic size in the security parameter.

Reducing Ciphertext Size. To further reduce the IKEMR ciphertext size for settings with frequent re-encapsulations, we add more TDPs to the IKEMR public key in order to avoid attaching epoch indices to re-encapsulated ciphertexts. Note that in our KEM-based IKEMR construction, each IKEMR public key consists of one f-sized batch of KEM public keys. For our above TDP-based IKEMR construction, these KEM public keys are replaced with TDP public keys. Now, for this final construction, each IKEMR public key consists of Δ batches of f TDP public keys. These additional batches are used when Charlie, at time t_2, wants to re-encapsulate a ciphertext originally created at time t_1, where $d = t_2 - t_1 \leq \Delta$. In this case, instead of directly re-encapsulating with the newest TDP public key and informing Bob about the re-encapsulation 'jump' from time t_1 to time t_2, Charlie continuously applies d TDP evaluations, one after another: from t_1 to $t_1 + 1$ to $t_1 + 2$, and so on until t_2. Thus, when Bob decapsulates, he can simply iterate over a continuous chain of trapdoor inversions in reverse. Only for larger re-encapsulation jumps of size $d > \Delta$, Charlie attaches epoch indexes to the ciphertext. Intuitively, if f stays roughly the same throughout the execution of the IKEMR, then public keys will be of size $O(f \cdot \Delta)$ and ciphertexts will be of size $O(\lambda + \log(\lambda) \cdot f/\Delta)$, regardless of the number of re-encapsulations; in particular, if $\Delta = f$, then ciphertexts will always be of size $O(\lambda)$, even after several re-encapsulations.

External Key Storage. Recall from above that the secret key of any IKEM must have size $f \cdot \lambda$, where f is the size of the current decapsulation interval. For long intervals or even intervals of dynamic size, this lower bound may induce an impractical storage overhead. Thus, we propose to split the secret key into two components—one small component and one larger component, of which only the former needs to be securely stored. The latter component, on the other hand, can be stored anywhere, and can even be publicly accessible. We furthermore desire a generic interface such that reads and writes to the original (virtual) secret key can still be efficiently performed. Indeed, such operations should only use the small securely-stored component, as well as small downloads from and uploads to the larger public component. Most importantly, we still want the security properties of IKEM to hold if the small securely-stored component is corrupted from time to time.

The notion of securely outsourcing a database with read- and write-access with *only FS* is very well-studied (e.g., [9,10,16], and references therein). Indeed, a construction for n-entry databases satisfying this notion is known such that the secret state is size $O(1)$ and the read/write overhead is $O(\log n)$. However, the security models from these prior works do not require secrecy of future writes if at any time the secret storage is corrupted. That is, to the best of our knowledge, no notion of securely outsourcing a database with *PCS* is known.

In Sect. 6, we introduce the notion of securely outsourcing a database with read- and write-access with *both* FS and PCS, and call it Interval RAM (IRAM). We then show that the construction mentioned above surprisingly satisfies this stronger notion of security. IRAM can be combined with our IKEM constructions to reduce the size of local secret storage, while maintaining security.

IKEMR vs. Proxy Re-Encryption. A primitive related to IKEMR is Proxy Re-Encryption (PRE). PRE extends standard public-key encryption with a key-based re-encryption mechanism: every secret key sk_A can compute a re-encryption key $rk_{A,B}$ for re-encryption to another public key pk_B. Using $rk_{A,B}$, ciphertext c_A, previously (re-)encrypted to public key pk_A, can be re-encrypted to c_B such that c_B behaves as if it was encrypted to pk_B without changing the payload.

PRE schemes can be bidirectional or unidirectional. With bidirectional PRE, a re-encryption key $rk_{A,B}$ for epochs A and B can be used to re-encrypt from A to B *and* vice versa. Thus, given sk_A and $rk_{A,B}$, there are naturally no security properties regarding ciphertexts encrypted to pk_B. For unidirectional PRE, re-encryption key $rk_{A,B}$ can *only* be used to re-encrypt from A to B, and given sk_A and $rk_{A,B}$, ciphertexts encrypted to pk_B are still secure (see, e.g., [20,23,28]).

Bidirectional PRE is clearly insufficient for secure IKEMR. Yet, IKEMR intuitively seems weaker than unidirectional PRE for three reasons (a formal lower bound is a harder task, which we deem out of scope): (1) In unidirectional PRE, re-encryption keys are only derived from the old secret key and the new *public* key; in IKEMR, the public keys used for re-encapsulation are derived from *both* old secret keys and the new secret key. (2) In IKEMR, ciphertexts can only be re-encapsulated to newer public keys; in unidirectional PRE, re-encryption keys can be derived for arbitrary public keys, which may even lead to full re-encryption circles. (3) Unidirectional PRE offers additional security guarantees if the re-encryption key remains secret—IKEMR public keys are, by definition, always public.

Indeed, we achieve IKEMR with constant-size ciphertexts from TDPs, while unidirectional PRE with constant-size ciphertexts can only be achieved from (expensive) FHE or iO currently (see, e.g., [28, p. 11]). Furthermore, even unidirectional PRE seems unsuitable for building IKEMR with FS directly: an unlimited chain of PRE re-encryption keys can be used to shift an old ciphertext to a much newer, corrupted secret key, which undermines FS.

Further Related Primitives. Like IKEM, primitives such as Updatable PKE (UPKE) [21,25,27] and Key-Updatable KEM (KU-KEM) [5,32,34] continuously update both parts of the key pair. Yet, these primitives only achieve FS and they are designed to work in a session-based fashion between a fixed tuple of Alice and Bob. Thereby, since the public keys in UPKE and KU-KEM are continuously updated by *Alice*, this would require synchronization when multiple *Alices* want to talk to the same Bob. IKEM overcomes this issue by letting all Alices use the newest public key they are aware of.

An orthogonal security feature that we do not cover in this work is in-epoch FS: the granularity of FS in IKEM is relatively coarse as only shrinking the interval by deleting entire epochs yields FS with respect to these deleted, old epochs. Using FS-KEM techniques—based on building blocks like identity-based encryption—, each epoch could have internal sub-steps for which FS can be achieved by updating Bob's secret key without invalidating the corresponding epoch entirely. We refrain from studying this aspect as it seems to be a simple, straight forward extension.

Active Security. For all our constructions, we prove security against active adversaries. Before we can do so, we develop a suitable security definition that models Chosen Ciphertext Attacks (CCA) by giving adversaries access to a decapsulation oracle. Upon a queried ciphertext c, this oracle honestly decapsulates the symmetric key k encapsulated in c using Bob's current secret key, unless c was posed as a *challenge*. Clearly, decapsulating the challenge ciphertext c^* via this oracle trivializes the adversary's ability to win the security experiment.

Due to re-encapsulations, muting the decapsulation oracle upon challenge ciphertexts is, however, non-trivial: the adversary can re-encapsulate challenge c^* using Bob's newer public keys, which yields c^{**}. Thus, the decapsulation oracle has to reject c^*, all of its re-encapsulations c^{**}, and so on. The literature on PRE developed several approaches for identifying such re-encapsulated challenges c^{**} of which we consider only one suitable: employing a Replayable CCA (RCCA) [18] definition style. However, the RCCA definition style is not directly applicable to KEM-type primitives. For this reason, we develop a suitable variant of RCCA security that is implied by CCA security for KEM-type primitives and can be used to build PKE that is RCCA-secure via standard hybrid encryption. We also show that our RCCA notion for IKEMR can be used to build the natural extension of this primitive that captures encryption, which we call *Interval Public Key Encryption with Re-encryptions* (IPKER). We elaborate on our definitional choices in the full version [8]. To add RCCA security to our constructions, we use standard techniques from the literature, which add minimal overhead.

We also extend our IRAM security notion to withstand active adversaries. Here, the receiver (with small local storage) must be able to detect if the adversary provides them with incorrect parts of the (larger) public storage for a given operation. To achieve this notion, we combine our original construction with techniques from the Memory Checking literature ([10,14,22] and the references therein), while adding minimal overhead.

Contributions. In summary, we develop a natural notion of KEM that offers FS and PCS guarantees against active adversaries. For this, we prove a lower bound in the full version [8] to show that the most basic construction is optimal. We extend the initial notion of IKEM by adding re-encapsulation, which we call IKEMR (see Sect. 3). Realizing IKEMR turns out to be more complicated: Our first construction in Sect. 4 uses Trapdoor Permutations to keep ciphertexts small. Our second construction in Sect. 5 uses FS-PKE to reduce the size of public keys. Finally, in Sect. 6, we introduce Interval RAM with which secret keys can be split into a locally stored part of constant size and a larger part that can be outsourced to an insecure (and actively adversarial) external storage.

2 Preliminaries

Below, we present some notation we will use throughout this work and then three important primitives which we will use in our constructions. In the full version [8], we provide some additional definitions for basic primitives used in our IKEM constructions, as well as some standard definitions and lemmas from information theory.

Notation. We use $x \leftarrow y$ for assigning value y to variable x. We use $x \leftarrow_{\$} \mathcal{X}$ to denote sampling x randomly from distribution $\mathcal{X}$. Consider some algorithm A. If A is deterministic, we use $y \leftarrow A(x)$ to denote assigning to y the output of $A(x)$. If A is randomized, we use $y \leftarrow_{\$} A(x)$ to denote assigning to y the output of a random run of $A(x)$. Sometimes, we may explicitly specify the random coins r that a randomized algorithm A uses; in this case, we use $y \leftarrow A(x; r)$ to denote assigning to y the output of a run of $A(x)$ using coins r.

Family of Lossy Trapdoor Permutations with a Common Domain. We now define families of lossy trapdoor permutations (TDPs), in which all permutations in the family share a common domain $\mathcal{X}$ [4]. Lossy TDPs can be instantiated in *injective* or *lossy* mode—in injective mode, every input $x \in \mathcal{X}$ permuted to $y \leftarrow \mathrm{P.eval}(pk, x)$ can be inverted back to $x \leftarrow \mathrm{P.inv}(sk, y)$; in lossy mode, the image of $\mathrm{P.eval}(pk, \cdot)$ is much smaller than $\mathcal{X}$ (and therefore, finding x from $\mathrm{P.eval}(pk, x)$ is statistically-hard). Furthermore, for any adversary with just the public key pk, it is hard to distinguish whether pk was sampled in injective or lossy mode.

Syntax. A family of *Lossy Trapdoor Permutations with Common Domain* (TDP) scheme P is a tuple of algorithms $\mathrm{P} = (\mathrm{P.gen}, \mathrm{P.eval}, \mathrm{P.inv})$ with the following syntax:

- $\mathrm{P.gen}(1^\lambda, b) \rightarrow_{\$} (sk, pk)$ generates a key-pair. Input $b \in \{0, 1\}$ specifies whether the generated instance is *injective* ($b = 1$) or *lossy* ($b = 0$).
- $\mathrm{P.eval}(pk, x) \rightarrow y$ takes in a public key pk and input x and permutes x to y.
- $\mathrm{P.inv}(sk, y) \rightarrow x$ takes in a secret key sk and permuted output y, and inverts it to x (looking ahead, when in lossy mode, there are no properties required).

Correctness. A family of lossy TDPs is *correct* if for any key pair sampled in *injective mode*, inputs x permuted to y can always be inverted to x.

Definition 1. *Scheme* P *is* correct *if for all* $(sk, pk) \leftarrow_{\$} \mathrm{P.gen}(1^\lambda, 1)$ *and* $x \in \mathcal{X}$, $x = \mathrm{P.inv}(sk, \mathrm{P.eval}(pk, x))$.

Security. For security of families of lossy TDPs, we require two properties. The first property is that when in lossy mode, the size of the image of $\mathrm{P.eval}(pk, \cdot)$ is much smaller than the domain $\mathcal{X}$.

Definition 2. *Scheme* P *is* L-lossy *if for all* $(sk, pk) \leftarrow_\$ \mathrm{P.gen}(1^\lambda, 0)$, $|\mathrm{P.eval}(pk, \cdot)| \leq |\mathcal{X}|/L$, *where* $|\mathrm{P.eval}(pk, \cdot)|$ *is the number of unique outputs across* $x \in \mathcal{X}$.

The second property is that an adversary that is given some sampled pk should not be able to tell if it was sampled in injective or lossy mode.

Definition 3. *Scheme* P *is* $(T, \varepsilon_\mathrm{P})$-secure *if for all adversaries* $\mathcal{A}$ *running in time* T*:* $\Pr[b \leftarrow_\$ \mathcal{A}(pk) : b \leftarrow_\$ \{0,1\}; (sk, pk) \leftarrow_\$ \mathrm{P.gen}(1^\lambda, b)] \leq 1/2 + \varepsilon_\mathrm{P}$.

Auerbach et al. show how to construct families of Lossy TDPs with a common domain $\mathcal{X} = \{0,1\}^n$ from many assumptions [4]. Of note for our purposes, they construct such a Lossy TDP family with lossiness $L = 2^{n/4}$ from the Phi-Hiding Assumption.

All-But-One Trapdoor Functions. We now define families of all-but-one (ABO) trapdoor functions, which are a generalization of lossy trapdoor functions. In an ABO family, each function has several *branches.* All of the branches are injective, except for one branch that is lossy. Moreover, an adversary with the public key pk of the ABO cannot tell which of the branches is lossy.

Syntax. A family of *all-but-one trapdoor functions* (ABO) scheme ABO is a tuple of algorithms ABO = (ABO.gen, ABO.eval, ABO.inv) with the following syntax:

– $\mathrm{ABO.gen}(1^\lambda, b) \rightarrow_\$ (sk, pk)$ generates a key-pair. Input $b \in \{0,1\}^v$, for $v \in \mathsf{poly}(\lambda)$ specifies the branch that is *lossy.*
– $\mathrm{ABO.eval}(pk, b, x) \rightarrow y$ takes in a public key pk, branch b, and input x and outputs y.
– $\mathrm{ABO.inv}(sk, b, y) \rightarrow x$ takes in a secret key sk, branch b, and output y, and inverts it to x (for lossy branch b, there are no properties required).

Correctness. A family of ABOs is *correct* if for any key pair, inputs x mapped to y for any *injective* branch can always be inverted to x (under the same branch).

Definition 4. *Scheme* ABO *is* correct *if for all* $b \neq b' \in \{0,1\}^v$, $(sk, pk) \leftarrow_\$ \mathrm{ABO.gen}(1^\lambda, b)$, *and* $x \in \mathcal{X}$, $x = \mathrm{ABO.inv}(sk, b', \mathrm{ABO.eval}(pk, b', x))$.

Security. For security of families of ABOs, we require two properties. The first property is that for the lossy branch b of the function, the size of the image of $\mathrm{ABO.eval}(pk, b, \cdot)$ is much smaller than the domain $\mathcal{X}$.

Definition 5. *Scheme* ABO *is* L-lossy *if for all* $b \in \{0,1\}^v$ *and* $(sk, pk) \leftarrow_\$ \mathrm{ABO.gen}(1^\lambda, b)$, $|\mathrm{ABO.eval}(pk, b, \cdot)| \leq |\mathcal{X}|/L$, *where* $|\mathrm{ABO.eval}(pk, b, \cdot)|$ *is the number of different outputs across all inputs* $x \in \mathcal{X}$.

The second property is that an adversary that is given some sampled pk should not be able to tell which branch is lossy.

Definition 6. *Scheme* ABO *is* $(T, \varepsilon_{\text{ABO}})$-secure *if for any* $b_0, b_1 \in \{0,1\}^v$*, for all adversaries* $\mathcal{A}$ *running in time* T*:* $\Pr[\delta \leftarrow_\$ \mathcal{A}(pk) : \delta \leftarrow_\$ \{0,1\}; (sk, pk) \leftarrow_\$ \text{ABO.gen}(1^\lambda, b_\delta)] \leq 1/2 + \varepsilon_{\text{ABO}}$.

Peikert and Waters, show how to construct a family of ABOs with domain $\mathcal{X} = \{0,1\}^n$ and lossiness $L = 2^n/\lambda$ from the DDH assumption [29].

Forward-Secure Public-Key Encryption. We briefly define FS-PKE [17], which is close to our definition of f-Bounded Forward-Secure Lossy TDPs with Common Domain in Sect. 4.

Syntax Forward-Secure Public-Key Encryption (FSE) is a tuple of algorithms FSE = (FSE.gen, FSE.up, FSE.enc, FSE.dec) with the following syntax:

- FSE.gen$(1^\lambda, f) \rightarrow_\$ (SK_0, PK)$ on input f that specifies the maximal number of update-epochs, outputs initial secret key SK_0 and public key PK.
- FSE.up$(SK_t) \rightarrow SK_{t+1}$ updates input secret key SK_t to SK_{t+1}.
- FSE.enc$(PK, t', m) \rightarrow_\$ c$ on input PK and epoch t', encrypts m for epoch t' in c.
- FSE.dec$(SK_t, t', c) \rightarrow m$ on input SK_t and t', decrypts c for epoch t' to output m.

Note that t' is an explicit input of FSE.dec(). For our usage of FSE.dec() within our IKEMR construction, we will be able to extract the proper t' from the rest of the ciphertext.

Correctness. Correctness requires that, for $t_0 \leq t_1$, the receiver with SK_{t_0} can decrypt $c \leftarrow \text{FSE.enc}(PK, t_1, m)$ to m.

Definition 7. *Scheme* FSE *is* correct *if for all* $(SK_0, PK) \leftarrow_\$ \text{FSE.gen}(1^\lambda, f)$*, for every* $t \in [f-1], SK_t \leftarrow \text{FSE.up}(SK_{t-1})$*, all* $m \in \mathcal{M}$ *and any* $0 \leq t_0 \leq t_1 \leq f-1$*,* $m = \text{FSE.dec}(SK_{t_0}, t_1, \text{FSE.enc}(PK, t_1, m))$*.*

Security. For any adversary with only the public key PK and *any* secret key $SK_{t'}$ for $t' > t^*$, we require that it is hard to tell which of two same-length messages was encrypted to epoch t^*:

Definition 8. *Scheme* FSE *is* $(T, \varepsilon_{\text{FSE}})$-secure *if for all adversaries* $\mathcal{A}$ *running in time* T*:*

$$\begin{aligned}\Pr[b \leftarrow_\$ \mathcal{A}^{\mathbf{Dec}^{\neq t^*}_{\neq c^*}}(SK_{t'}) : b \leftarrow_\$ \{0,1\}; f \leftarrow_\$ \mathcal{A}(); (SK_0, PK) \leftarrow_\$ \text{FSE.gen}(1^\lambda, f);\\ (0 \leq t^* \leq f-1, m_0, m_1) \leftarrow_\$ \mathcal{A}^{\mathbf{Dec}}(PK); |m_0| = |m_1|;\\ c^* \leftarrow_\$ \text{FSE.enc}(PK, t^*, m_b); (t^* < t' \leq f) \leftarrow_\$ \mathcal{A}^{\mathbf{Dec}^{\neq t^*}_{\neq c^*}}(c^*);\\ \textit{for } t \in [f-1], SK_t \leftarrow \text{FSE.up}(SK_{t-1})] \leq 1/2 + \varepsilon_{\text{FSE}},\end{aligned}$$

where decryption oracle $\mathbf{Dec}^{\neq t^*}_{\neq c^*}$ *on input* (t, t°, c) *outputs* $\text{FSE.dec}(SK_t, t^\circ, c)$*, unless* $t \leq t^\circ = t^*$ *and* $c = c^*$*.*

3 Interval KEM with Re-Encapsulations

In this section, we introduce our Interval Key-Encapsulation Mechanism with Re-Encapsulations (IKEMR) notion. We provide a security definition with a decryption oracle and Replayable CCA-style security,[2] but also, by simply removing the decryption oracle, provide a CPA-style security notion. We also explain that the secret state of IKEMR schemes must be large (following from a lower bound of [19] for a different, simpler, symmetric-key primitive). Later, we will present two different constructions for this IKEMR notion that provide incomparable efficiency properties. We begin by defining the IKEMR notion:

Syntax. An *Interval Key-Encapsulation Mechanism with Re-Encapsulations* (IKEMR) scheme IKMR is a tuple of algorithms IKMR = (IKMR.gen, IKMR.enc, IKMR.dec, IKMR.re-gen, IKMR.del, IKMR.re-enc) with the following syntax:

- IKMR.gen$(1^\lambda) \rightarrow_\$ (st_{0,0}, pk_0)$ generates a secret state and a corresponding public key for interval $[t_0, t_1]$ with $t_0 = t_1 = 0$.
- IKMR.re-gen$(st_{t_0,t_1}) \rightarrow_\$ (st_{t_0,t_1+1}, pk_{t_1+1})$ updates the secret state st_{t_0,t_1} to st_{t_0,t_1+1}, and outputs fresh public key pk_{t_1+1}; i.e., starting new epoch $t_1 + 1$ and setting $t_1 \leftarrow t_1 + 1$.
- IKMR.del$(st_{t_0,t_1}, \ell) \rightarrow st_{t_0+\ell,t_1}$ on input secret state st_{t_0,t_1}, deletes from the secret state the material needed to decapsulate keys encapsulated in the epochs $[t_0, t_0 + \ell)$ and outputs $st_{t_0+\ell,t_1}$; i.e., setting $t_0 \leftarrow t_0 + \ell$.
- IKMR.enc$(pk_{t_1}) \rightarrow_\$ (k, c)$ on input public key pk_{t_1}, encapsulates key k in ciphertext c.
- IKMR.dec$(st_{t_0,t_1}, c) \rightarrow k$ on input secret state st_{t_0,t_1} and ciphertext c, decapsulates key k.
- IKMR.re-enc$(pk_{t_1}, c) \rightarrow_\$ c'$ re-encapsulates input ciphertext c with respect to the input public key pk_{t_1}.

Correctness. An IKEMR scheme is *correct* if secret state st_{t_0,t_1} can decapsulate correctly any ciphertext that was *originally* created in any epoch $t \in [t_0, t_1]$. In particular, even if some ciphertext is re-encapsulated during epoch $t' \in [t_0, t_1]$, if the epoch in which the ciphertext was *originally* created (i.e., when IKMR.enc was executed) is $t < t_0$, then no correctness is required. In fact, as we will see below, such ciphertexts must be secure even given st_{t_0,t_1}. More formally:

Definition 9. *Given $T \in \mathbb{N}$, and dictionary D s.t. $D[i] = (t^i, \ell_i)$ for $i \in [m]$ containing items in $[T] \times [T]$ s.t. $0 < t^1 \leq \cdots \leq t^m \leq T$ and $\sum_{j=1}^{i} \ell_j \leq t^i$ for every $i \in [m]$: let $t_0 \leftarrow 0$; $(st_{0,0}, pk_0) \leftarrow_\$$* IKMR.gen(1^λ)*; and for all $t_1 \in [T]$, $(st_{t_0,t_1}, pk_{t_1}) \leftarrow_\$$* IKMR.re-gen(st_{t_0,t_1-1}) *and for every $i \in [m]$ s.t. for $(t^i, \ell_i) \leftarrow D[i]$, $t^i = t_1$, $st_{t_0+\ell_i,t_1} \leftarrow$* IKMR.del$(st_{t_0,t_1}, \ell_i)$. *Scheme* IKMR *is*

[2] See Sect. 1 and the full version [8] for elaboration on this choice, mainly stemming from the problem of handling decryptions of honest re-encapsulations of the challenge ciphertext.

correct *if for every such T and D, as well as $r \leq t_1 - t_0 + 1$, $R = \{t_\rho^1, \dots, t_\rho^r\} \subseteq [t_0, t_1]$ s.t. $t_\rho^1 < \cdots < t_\rho^r$, $(k, c_1) \leftarrow_\$$ IKMR.enc$(pk_{t_\rho^1})$, and for $i \in [2, r]$, $c_i \leftarrow_\$$ IKMR.re-enc$(pk_{t_\rho^i}, c_{i-1})$, $k =$ IKMR.dec(st_{t_0,t_1}, c_r).*

Initialization: Set $t_X^0, t_X^1, t_R, t^* \leftarrow -\infty$, $t_0, t_1 \leftarrow 0$, and $\mathit{pub\text{-}chall} \leftarrow 0$. Then compute $(st_{t_0,t_1}, pk_{t_1}) \leftarrow_\$ \text{IKMR.gen}(1^\lambda)$ and output pk_{t_1}.

```
Re-Gen():
  1. increment $t_1 \leftarrow t_1 + 1$
  2. regenerate $(st_{t_0,t_1}, pk_{t_1}) \leftarrow_\$$
       IKMR.re-gen$(st_{t_0,t_1-1})$
  3. return $pk_{t_1}$

Chall(pub):
  1. return $\perp$ if $pub = 1$ and $t_1 = t_X^1$
  2. set $t_R, t^* \leftarrow t_1$
  3. encapsulate $(k_0, c^*) \leftarrow_\$$
       IKMR.enc$(pk_{t_1})$
  4. sample random $k_1 \leftarrow_\$ \mathcal{K}$
  5. flip random coin $b \leftarrow_\$ \{0, 1\}$
  6. disable Chall()
  7. if $pub = 1$ set $\mathit{pub\text{-}chall} \leftarrow 1$
       and return $((k_b, k_{1-b}), c^*)$

Expose():
  1. return $\perp$ if $\mathit{pub\text{-}chall} = 1$ and $t^* \geq t_0$
  2. if $t_1 > t_X^1$: set $t_X^0 \leftarrow t_0, t_X^1 \leftarrow t_1$
  3. return $st_{t_0,t_1}$

Re-Enc-Chall(pub):
  1. return $\perp$ if
     (a) $\mathit{pub\text{-}chall} = 1$; or
     (b) $t^* = -\infty$; or
     (c) $t_1 = t_R$; or
     (d) $pub = 1$, $t_1 = t_X^1$ and $t^* \geq t_X^0$
  2. set $t_R \leftarrow t_1$
  3. re-encapsulate $c^* \leftarrow_\$$
       IKMR.re-enc$(pk_{t_1}, c^*)$
  4. if $pub = 1$: set $\mathit{pub\text{-}chall} \leftarrow 1$
       and return $((k_b, k_{1-b}), c^*)$

Dec(c):
  1. decapsulate $k \leftarrow$ IKMR.dec$(st_{t_0,t_1}, c)$
  2. return $\perp$ if $k \in \{k_0, k_1\}$
  3. return $k$

Del(ℓ):
  1. return $\perp$ if $t_0 + \ell > t_1$
  2. increment $t_0 \leftarrow t_0 + \ell$
  3. compute $st_{t_0,t_1} \leftarrow$
       IKMR.del$(st_{t_0-\ell,t_1}, \ell)$
```

Fig. 2. IKEMR IND-X$_{\text{IKMR}}$ security game, for $\text{X} \in \{\text{CPA}, \text{RCCA}\}$. Components only needed for the IND-RCCA$_{\text{IKMR}}$ security game are written in green. (Color figure online)

Security. Now we define security for an IKEMR scheme. At a high level, this security (i) allows for the challenge ciphertext encrypted in epoch t^* to not be made public (i.e., unavailable to the adversary by setting $pub = 0$) immediately, (ii) allows for re-encapsulations of the challenge ciphertext before it is made public, and (iii) is required if and only if once the challenge ciphertext is made public (via $pub = 1$) after a re-encapsulation in epoch t_1, for every state $st_{t_0',t_1'}$ that the adversary had exposed before the time of publication, either $t^* < t_0'$ or $t_1 > t_1'$ (and also the adversary waits until $t_0 > t^*$ before exposing the state again). In particular, the last condition of the last item implies that even if the adversary earlier exposed multiple states $st_{t'_0,t'_1}$ such that $t^* \in [t'_0, t'_1]$, *before the challenge ciphertext was made public*, then if the receiver re-generates its state to output a fresh public key, is not exposed again until after $t_0 > t^*$, and the challenge ciphertext is re-encrypted with respect to the above public key, the new ciphertext is required to be secure. Our indistinguishability notion is mildly atypical as, once the challenge ciphertext is made public, the adversary is given either (with equal probability 1/2) (i) the real key k_0 encapsulated by the challenge ciphertext, followed by a random key k_1, i.e., (k_0, k_1); or (ii) (k_1, k_0). The adversary must guess if they are in world (i) or (ii). Additionally, for Replayable CCA (RCCA) security, the adversary is allowed to see decryptions of ciphertexts of its choosing, as long as they do not decrypt to k_0 or k_1. In the full version [8], we provide justification for this slightly modified RCCA definition; e.g., it implies RCCA-secure IPKER.

More formally, we define the IND-X_{IKMR} ($\text{X} \in \{\text{CPA}, \text{RCCA}\}$) security game for IKEMR in Fig. 2. The security game starts by sampling a key pair via IKMR.gen and returning the public key to the adversary. It also initializes a number of variables, including $t_X^0, t_X^1 \leftarrow -\infty$, which will be used to store the endpoints of the latest state $st_{t_X^0,t_X^1}$ that the adversary exposed (these endpoints are only updated when $t_1 > t_X^1$). In addition, the game keeps track of the latest epoch t_R in which the challenge ciphertext was (re-)encapsulated, as well as *pub-chall* which indicates if the challenge ciphertext has been made public (set to 1 if so). Oracle **Re-Gen**() re-generates the state using IKMR.re-gen() and returns the new public key. **Del**(ℓ) deletes old key material for the last ℓ epochs from the state using IKMR.del($\cdot, \ell$), only if the updated interval would still be valid; i.e., $t_0 + \ell \leq t_1$. Oracle **Chall**(*pub*), if $pub = 1$, first checks if the receiver's state has not been exposed in the current epoch. If not, it runs IKMR.enc() to obtain challenge ciphertext c^* and encapsulated key k_0, then samples random k_1 and bit b, and finally returns $((k_b, k_{1-b}), c^*)$. If $pub = 0$, then no matter what, **Chall**() runs IKMR.enc(), but does not output the challenge ciphertext c^* or keys (k_0, k_1); only stores them as well as the challenge epoch t^* and $t_R \leftarrow t^*$. In both cases, the **Chall** oracle is thereafter disabled. Oracle **Expose**() always returns the current state if the challenge ciphertext has not been made public yet (*pub-chall* $= 0$). Otherwise, if the challenge ciphertext is public, **Expose**() first checks that the challenge epoch t^* is not at least t_0, the lower endpoint for which the current state remembers key material, since then the adversary can trivially decapsulate the challenge ciphertext and win the game.

Finally, the re-encapsulation oracle: **Re-Enc-Chall**(*pub*). This oracle first returns $\perp$ if (i) the challenge ciphertext has already been made public (*pub-chall* $= 1$), (ii) the challenge ciphertext has not already been created ($t^* = -\infty$), or (iii) **Re-Gen**() has not been queried since the last **Chall**() or **Re-Enc-Chall**() query. Now, if $pub = 0$, and the above checks have passed, then **Re-Enc-Chall**() runs $c_1^* \leftarrow_\$ \text{IKMR.re-enc}(\cdot, c_0^*)$, stores the re-encapsulated ciphertext c_1^*, and updates $t_R \leftarrow t_1$ to the current epoch, but does not output anything. If $pub = 1$, then in addition to the above checks, **Re-Enc-Chall**()

aborts if for the latest state $st_{t_X^0,t_X^1}$ that the adversary exposed, $t_X^1 = t_1$ (the current epoch) and *original* challenge epoch $t^* \geq t_X^0$, since then the adversary could trivially decapsulate the challenge ciphertext and win the game. Once the checks have passed, **Re-Enc-Chall**() runs $c_1^* \leftarrow_\$ \text{IKMR.re-enc}(\cdot, c_0^*)$, sets *pub-chall* $\leftarrow 1$ and outputs $((k_b, k_{1-b}), c_1^*)$, where k_0, k_1 are the real and random keys.

For RCCA security, there is also the **Dec**(c) oracle, written in green in Fig. 2. This oracle uses the current state st_{t_0,t_1} to decrypt c using IKMR.dec() and returns the resulting key k only if $k \notin \{k_0, k_1\}$. Given this security game, we now formally define secure IKEMR schemes:

Definition 10. *For* $\text{X} \in \{\text{CPA}, \text{RCCA}\}$*, an IKEMR scheme* IKMR *is* $(T, \varepsilon^{\text{ind-x}}_{\text{IKMR}})$-secure *if for all adversaries* $\mathcal{A}$ *playing the security game* $\text{IND-X}_{\text{IKMR}}$ *and running in time* T*:* $\Pr[b \leftarrow_\$ \text{IND-X}_{\text{IKMR}}(\mathcal{A})] \leq 1/2 + \varepsilon^{\text{ind-x}}_{\text{IKMR}}$.

Lower Bound on Secret State Size. Unfortunately, we prove a lower bound that shows the size of the secret state of any IKEMR scheme must be proportional to the current interval size, $t_1 - t_0$ (times the security parameter, λ). As we write in Sect. 1, one way to show this would be to prove that IKEMR implies a different and simpler symmetric-key primitive, called *Self Encrypted Queue* introduced by Choi et al. [19], for which they also prove a corresponding lower bound. Nevertheless, we provide in the full version [8] a direct lower bound proof showing that the IKEMR secret state size must be $\Omega(\lambda \cdot (t_1 - t_0))$, as it more clearly illustrates the intuition behind why this is the case. Moreover, the lower bound holds for the simpler IKEM notion without re-encapsulations.

4 IKEMRConstruction from Lossy TDPs with Common Domain

We now present our IKEMR construction from Lossy TDPs with Common Domain that matches the lower bound on secret state size mentioned above. This construction optimizes for small ciphertexts, as small as $O(\lambda + \log(T))$ bits, where T is the total number of epochs, even after many re-encryptions. We provide a RCCA-secure construction that additionally makes use of all-but-one trapdoor functions and one-time signatures (based on a technique of [29]). We also demonstrate that by simply removing the ABO and OTS, we obtain a construction that is CPA-secure.

As an intermediate building block, we first define *f-Bounded Forward-Secure Lossy Trapdoor Permutations with Common Domain* and instantiate it using Lossy TDPs with Common Domain.

4.1 *f*-Bounded Forward-Secure Lossy Trapdoor Permutation with Common Domain

We now introduce f-Bounded Forward-Secure Lossy TDPs (FS-TDPs) with Common Domain. This primitive is similar to f-bounded Forward-Secure KEMs

[17], except we require security properties corresponding to Lossy TDPs, instead of KEMs.

Syntax. A family of *f-Bounded Forward-Secure Lossy TDPs with Common Domain* (FSP) is a tuple of algorithms FSP $=$ (FSP.gen, FSP.up, FSP.eval, FSP.inv) with the following syntax:

- FSP.gen$(1^\lambda, f, b, t^*) \rightarrow_\$ (SK_0, PK)$ on inputs $b \in \{0,1\}$ and t^* that specify whether the generated instance is *always-injective* ($b = 1$) or *lossy* in epoch t^* ($b = 0$), outputs initial secret key SK_0 and public key PK.
- FSP.up$(SK_t) \rightarrow SK_{t+1}$ updates input secret key SK_t to SK_{t+1}.
- FSP.eval$(PK, t', x) \rightarrow y$ on input PK and epoch t', evaluates the permutation for epoch t' on x to give output y.
- FSP.inv$(SK_t, t', y) \rightarrow x$ on input SK_t and t', inverts the permutation for epoch t' on y to give x.

Note that t' is an explicit input of FSP.inv(). For our use of FSP.inv() within our IKEMR construction, we will be able to extract t' from the rest of the ciphertext.

Correctness. Correctness requires that in *always-injective* mode, for $t_0 \leq t_1$, the receiver with SK_{t_0} can invert $y \leftarrow$ FSP.eval(PK, t_1, x) to x.

Definition 11. *Scheme* FSP *is* correct *if for all* $(SK_0, PK) \leftarrow_\$$ FSP.gen$(1^\lambda, f, 1, t^*)$*, for every* $t \in [f-1]$, $SK_t \leftarrow$ FSP.up(SK_{t-1})*, all* $x \in \mathcal{X}$ *and any* $0 \leq t_0 \leq t_1 \leq f-1$, $x =$ FSP.inv$(SK_{t_0}, t_1,$ FSP.eval$(PK, t_1, x))$.

Security. For *lossy* mode with respect to given epoch $t^* < f$, we require that the size of the image of the evaluation algorithm with respect to epoch t^* is much smaller than the domain $\mathcal{X}$:

Definition 12. *Scheme* FSP *is* L-lossy *if for all* $(SK_0, PK) \leftarrow_\$$ FSP.gen$(1^\lambda, f, 0, t^*)$, $|$FSP.eval$(PK, t^*, \cdot)| \leq |\mathcal{X}|/L$*, where* $|$FSP.eval$(PK, t^*, \cdot)|$ *is the number of different outputs across all inputs* $x \in \mathcal{X}$*.*

Furthermore, for any adversary with only the public key PK and *any* secret key $SK_{t'}$ for $t' > t^*$, we require that it is hard to distinguish whether they were sampled in always-injective mode or lossy mode with respect to t^* (even with t^* known to the adversary):

Definition 13. *Scheme* FSP *is* $(T, \varepsilon_{\mathrm{FSP}})$-secure *if for all adversaries* $\mathcal{A}$ *running in time* T*:*

$$\begin{aligned}\Pr[b \leftarrow_\$ \mathcal{A}(SK_{t'}) : b \leftarrow_\$ \{0,1\}; (0 \leq t^* \leq f-1) \leftarrow_\$ \mathcal{A}();\\ (SK_0, PK) \leftarrow_\$ \mathrm{FSP.gen}(1^\lambda, f, b, t^*); (t^* < t' \leq f) \leftarrow_\$ \mathcal{A}(PK);\\ \textit{for } t \in [f-1], SK_t \leftarrow \mathrm{FSP.up}(SK_{t-1})] \leq 1/2 + \varepsilon_{\mathrm{FSP}}\end{aligned}$$

FSP Construction. We now provide a simple construction based on a family of Lossy TDPs with Common Domain P and PRG G. At a high level, FSP.gen will generate f instantiations of P: for epoch t^* and lossiness bit b, the t^*-th instantiation of P will be generated with bit b; for all other epochs t, the t-th instantiation of P will be generated with bit 1 (i.e., injective). The initial secret key SK_0 will store a PRG s_0 which can be expanded deterministically (in a chain) to sample f secret keys and the public key PK will store the f corresponding public keys. To update secret key SK_t, the receiver simply expands the seed s_t and deletes the output secret key corresponding to the t-th instantiation of P (while still s_{t+1} can be expanded to compute all future secret keys). To evaluate with respect to epoch t, the sender simply evaluates the t-th instantiation of P on x. Finally, to invert using SK_t on input y and epoch t', the receiver simply expands s_t iteratively to get sk_t' corresponding to the t'-th instantiation of P and uses it to invert y. The scheme is as follows:

- FSP.gen$(1^\lambda, f, b, t^*)$: set $t \leftarrow 0$ and sample random $s_0 \leftarrow_\$ \mathcal{S}$. For $i \in [0, f-1] \setminus \{t^*\}$, compute $(s_{i+1}, r_i) \leftarrow \mathrm{G}(s_i)$ and sample $(sk_i, pk_i) \leftarrow_\$ \mathrm{P.gen}(1^\lambda, 1; r_i)$. For t^*, compute $(s_{t^*+1}, r_{t^*}) \leftarrow \mathrm{G}(s_{t^*})$ and sample $(sk_{t^*}, pk_{t^*}) \leftarrow_\$ \mathrm{P.gen}(1^\lambda, b; r_{t^*})$. Set $SK_t \leftarrow (t, s_0)$ and output $PK \leftarrow \{pk_0, \ldots, pk_{f-1}\}$.
- FSP.up(SK_t): Compute $(s_{t+1}, \cdot) \leftarrow \mathrm{G}(s_t)$ and set $t \leftarrow t + 1$.
- FSP.eval(PK, t, x): Compute and output $y \leftarrow \mathrm{P.eval}(pk_t, x)$.
- FSP.inv(SK_t, t', y): For i from t to t': compute $(s_{i+1}, r_i) \leftarrow_\$ \mathrm{G}(s_i)$. Then sample $(sk_{t'}, \cdot) \leftarrow_\$ \mathrm{P.gen}(1^\lambda, 1; r_{t'})$ and output $x \leftarrow \mathrm{P.inv}(sk_{t'}, y)$.

We now show that the above FSP construction is correct, L-lossy, and secure. The correctness and L-lossiness clearly follows from that of P. Furthermore, for security, since the FSP secret key $SK_{t'}$ that the adversary receives will not contain the lossy epoch t^*'s secret key sk_{t^*} of the lossy TDP family P (as it will have been deleted), but only a random PRG seed past epoch t^* in the chain, security follows directly from that of P and G. The proof of the following Theorem is provided in the full version [8].

Theorem 1. *If* P *is correct, L-lossy, and $(T, \varepsilon_{\mathrm{P}})$-secure, and* G *is $(T, \varepsilon_{\mathrm{G}})$-secure then the above* FSP *construction is correct, L-lossy, and $(T', \varepsilon_{\mathrm{P}} + T \cdot \varepsilon_{\mathrm{G}})$-secure, for $T' \approx T$.*

4.2 IKMR Construction

Given the FSP primitive, we can now present our construction for IKEMR. The construction IKMR_Δ is formally presented in Fig. 3. It is parameterized by an (efficiently computable) function $\Delta : \mathbb{N} \to \mathbb{N}$, such that for each input T, $\Delta(T) \leq T$. Intuitively, if the size of the active interval $t_1 - t_0$ stays roughly the same throughout, then the public key size will be proportional to $\Delta(t_1 - t_0)$ and the size of any ciphertext (regardless of the number of times it's been (re-)encapsulated), will be proportional to $\lambda + \beta \cdot (t_1 - t_0)/\Delta(t_1 - t_0)$, where $\beta = O(\log \lambda)$ is the number of bits needed to represent each epoch. In particular, if $\Delta(T) = T$, then all ciphertexts (no matter how many re-encapsulations) will be of size proportional to only λ (since $\beta = O(\log \lambda)$). Moreover, even if $\Delta(T)$ is small (even $\Delta(T) = 1$), then the ciphertext grows with (almost) every re-encapsulation, but only by a β factor, *independent* of λ.

In addition to the FSP primitive, our construction IKMR_{Δ} utilizes a family $\mathcal{H}$ of pairwise independent hash functions from $\{0,1\}^n \rightarrow \{0,1\}^{\ell}$. For RCCA security, $\ell \leq k - 2\log(1/\varepsilon_{\mathcal{H}})$, for some $k = \omega(\log n)$ and negligible $\epsilon_{\mathcal{H}} = \mathsf{negl}(\lambda)$; for CPA security, $\ell \leq \log(L) - 2\log(1/\epsilon_{\mathcal{H}})$, where L the lossiness of FSP. Additionally for RCCA security, we make use of an all-but-one trapdoor function family ABO and one-time signature scheme OTS.

In IKMR_{Δ}.gen, the receiver samples random hash function $h \leftarrow_{\$} \mathcal{H}$, generates an FSP instance $(sk_1, pk_1) \leftarrow_{\$} \text{FSP.gen}(1^{\lambda}, t_1 - t_0 + 1, 1, \perp)$, sets $t_0, t_1 \leftarrow 1$, and sets $(SK[t_1], PK[t_1]) \leftarrow ((sk_1, t_0), (pk_1, t_0))$. We will explain the choice of instantiating FSP with $f = t_1 - t_0 + 1$ while explaining IKMR_{Δ}.re-gen and IKMR_{Δ}.del below. Additionally, for RCCA security, the receiver samples an ABO instance $(\cdot, pk) \leftarrow_{\$} \text{ABO.gen}(1^{\lambda}, 0^v)$, where v is the bit-length of verification keys generated by OTS (ABO is only needed for security and indeed the secret key of the ABO is not used by the receiver). For each IKMR_{Δ}.re-gen execution, the receiver increments $t_1 \leftarrow t_1 + 1$, samples a new FSP instance $(sk_{t_1}, pk_{t_1}) \leftarrow_{\$} \text{FSP}(1^{\lambda}, t_1 - t_0 + 1, 1, \perp)$, sets $(SK[t_1], PK[t_1]) \leftarrow ((sk_{t_1}, t_0), (pk_{t_1}, t_0))$, and deletes from PK all entries except those for the latest $\Delta(t_1 - t_0) + 1 \leq t_1 - t_0 + 1$ epochs. The reason we instantiate the FSP with $f = t_1 - t_0 + 1$ is that we will use the FSP epoch $t' \in [0, f-1]$ to (re-)encapsulate IKEMR ciphertexts originally created in IKEMR epoch $t_0 + t' \in [t_0, t_1]$ (explanation continues after IKMR_{Δ}.del below). Then in $\text{IKMR}_{\Delta}.\text{del}(\ell)$, the receiver simply deletes from the secret key SK the FSP keys $sk_{t_0}, \ldots, sk_{t_0+\ell-1}$, and updates all other FSP keys $sk_{t_0+\ell}, \ldots, sk_{t_1}$, ℓ times each. Therefore, even if $\text{IKMR}_{\Delta}.\text{del}(\ell_i)$ is called m times such that $\sum_{i=1}^{m} \ell_i \leq t'$, the lower endpoint of the active interval is moved from t_0 to at most $t_0 + t'$. Thus, the FSP instance sampled when the lower endpoint was t_0, having been updated at most t' times will still be able to decapsulate IKEMR ciphertexts originally created in IKEMR epoch $t_0 + t'$. However, if $\sum_{i=1}^{m} \ell_i > t'$, moving the lower endpoint of the active interval from t_0 to after $t_0 + t'$, then the FSP instance sampled when the lower endpoint was t_0, having been updated more than t' times will no longer be able to decapsulate IKEMR ciphertexts originally created in IKEMR epoch $t_0 + t'$, as required by security.

Thus, to encapsulate in epoch t_1, IKMR_{Δ}.enc samples random $x \leftarrow_{\$} \{0,1\}^n$, then evaluates the FSP for public key pk_{t_1} on x and the FSP instantiation's epoch $t_1 - t_0$ to get output c_1. For RCCA security, the encapsulator also (i) samples OTS key pair (sk, vk); (ii) evaluates the ABO on branch vk and input x to obtain output c_2; and (iii) finally signs c_2 using OTS with signing key sk to obtain σ. The output key is $h(x)$ and the ciphertext is (c_1, vk, c_2, σ) appended with tuple (t_1, t_1). Note that in the RCCA-secure construction, vk, c_2, σ will remain untouched in the ciphertext even after re-encapsulations.

To re-encapsulate ciphertext $(c_1, vk, c_2, \sigma, ((t_{0,0}, t_{0,1}), \ldots, (t_{l-1,0}, t_{l-1,1}), (t_{l,0}, t_{l,1})))$ in epoch t_1 on input public key with active interval $[t_0, t_1]$, $t_{0,0}$ is inter-

```
IKMR_Δ.gen(1^λ):
  1. set $t_0, t_1 \leftarrow 1$, $SK[\cdot], PK[\cdot] \leftarrow \perp$
  2. sample $h \leftarrow_\$ \mathcal{H}$
  3. generate $(sk, pk) \leftarrow_\$$ FSP.gen$(1^\lambda, 1, 1, \perp)$
  4. generate $(\cdot, pk') \leftarrow_\$$ ABO.gen$(1^\lambda, 0^v)$
  5. set $(SK[1], PK[1]) \leftarrow ((sk, t_0), (pk, t_0))$
  6. return $((SK, PK, pk', h, t_0, t_1), (PK, pk', h, t_0, t_1))$

IKMR_Δ.del((SK, PK, pk', h, t_0, t_1), ℓ):
  1. for $i \in [\ell]$:
     (a) set $SK[t_0 + i - 1] \leftarrow \perp$
     (b) for $t \in [t_0 + \ell, t_1]$:
           set $(sk, t') \leftarrow SK[t]$;
           update $sk' \leftarrow$ FSP.up$(sk)$;
           store $SK[t] \leftarrow (sk', t')$
  2. increment $t_0 \leftarrow t_0 + \ell$
  3. return $(SK, PK, pk', h, t_0, t_1)$

IKMR_Δ.re-enc((PK, pk', h, t_0, t_1),
    (c_1, vk, c_2, σ,
    ((t_{0,0}, t_{0,1}), ..., (t_{l,0}, t_{l,1})))):
  1. if $t_0 > t_{0,0}$ return $\perp$
  2. if $PK[t_{l,1} + 1] \neq \perp$:
     (a) for $t'$ from $(t_{l,1} + 1)$ to $t_1$:
           let $(pk_{t'}, t'_0) \leftarrow PK[t']$;
           compute $c_1 \leftarrow$
             FSP.eval$(pk, t_{0,0} - t'_0, c_1)$
     (b) return $(c_1, vk, c_2, \sigma, ((t_{0,0}, t_{0,1}), \dots, (t_{l,0}, t_1)))$
  3. else:
     (a) compute $c_1 \leftarrow$
           FSP.eval$(PK[t_1], t_{0,0} - t_0, c_1)$
     (b) return $(c_1, vk, c_2, \sigma, ((t_{0,0}, t_{0,1}), \dots, (t_{l,0}, t_{l,1}), (t_1, t_1)))$

IKMR_Δ.re-gen((SK, PK, pk', h, t_0, t_1)):
  1. increment $t_1 \leftarrow t_1 + 1$
  2. generate $(sk, pk) \leftarrow_\$$ FSP.gen$(1^\lambda, t_1 - t_0 + 1, 1, \perp)$
  3. set $(SK[t_1], PK[t_1]) \leftarrow ((sk, t_0), (pk, t_0))$
  4. set $PK[\leq t_1 - \Delta(t_1 - t_0) - 1] \leftarrow \perp$
  5. return $((SK, PK, pk', h, t_0, t_1), (PK, pk', h, t_0, t_1))$

IKMR_Δ.enc((PK, pk', h, t_0, t_1)):
  1. sample random $x \leftarrow_\$ \mathcal{X}$
  2. compute $c_1 \leftarrow$ FSP.eval$(PK[t_1], t_1 - t_0, x)$
  3. generate $(sk, vk) \leftarrow_\$$ OTS.gen$(1^\lambda)$
  4. compute $c_2 \leftarrow$ ABO.eval$(pk', vk, x)$
  5. sign $\sigma \leftarrow$ OTS.sign$(sk, c_2)$
  6. return $(h(x), (c_1, vk, c_2, \sigma, ((t_1, t_1))))$

IKMR_Δ.dec((SK, PK, pk', h, t_0, t_1),
    (c_1, vk, c_2, σ,
    ((t_{0,0}, t_{0,1}), ..., (t_{l,0}, t_{l,1})))):
  1. return $\perp$ if OTS.ver$(vk, c_2, \sigma) = 0$
  2. set $c'_1 \leftarrow c_1$
  3. for $i$ from $l$ to 0:
     (a) for $t'$ from $t_{i,1}$ to $t_{i,0}$: let
           $(sk_{t'}, t'_0) \leftarrow SK[t']$; invert
           $c'_1 \leftarrow$ FSP.inv$(SK[t'], t_{0,0} - t'_0, c'_1)$
  4. set $x \leftarrow c'_1$
  5. for $i$ from 0 to $l$
     (a) for $t'$ from $t_{i,0}$ to $t_{i,1}$: let
           $(pk_{t'}, t'_0) \leftarrow PK[t']$;
           compute $c'_1 \leftarrow$
             FSP.eval$(PK[t'], t_{0,0} - t'_0, c'_1)$
  6. return $\perp$ if $c'_1 \neq c_1$
  7. compute $c'_2 \leftarrow$ ABO.eval$(pk', vk, x)$
  8. return $\perp$ if $c'_2 \neq c_2$
  9. compute and return $h(x)$
```

Fig. 3. TDP-based IKEM with Re-Encapsulations construction. Text written in greenis only needed for IND-RCCA$_{\text{IKMR}}$ security. (Color figure online)

preted as the epoch in which the ciphertext was originally created. Thus, IKMR_Δ.re-enc first returns $\perp$ if $t_0 > t_{0,0}$ since if this is the case, then the receiver must not be able to decapsulate the ciphertext anyway, as required for security. For the RCCA-secure construction, the re-encapsulator also keeps vk, c_2, σ untouched in the eventually output ciphertext. Then, IKMR_Δ.re-enc checks if there is an FSP public key in PK for epoch $t_{l,1}+1$. If so, then for t' from $t_{l,1}+1$ to t_1: IKMR_Δ.re-enc first retrieves $(pk_{t'}, t'_0) \leftarrow PK[t']$, where t'_0 was the lower endpoint of the active interval *when epoch t' was created.* Next IKMR_Δ.re-enc evaluates the FSP for public key $pk_{t'}$ on c_1 and the FSP instantiation's epoch $t_{0,0} - t'_0$, as specified above, to get new output c'_1. Then, IKMR_Δ.re-enc outputs the same ciphertext as above, except c_1 replaced by the final output c'_1 and $t_{l,1}$ of the final evaluation epoch tuple replaced with t_1. The latter is because each of these tuples represent the evaluation intervals of epochs t' in which the FSP for the corresponding public key $pk_{t'}$ was evaluated on c.

If there is no FSP public key in PK for epoch $t_{l,1}+1$, then IKMR_Δ.re-enc just evaluates the FSP for public key pk_{t_1} of epoch t_1 on c_1 and the FSP instantiation's epoch $t_{0,0} - t_0$, as specified above, to get new output c'_1. Then, IKMR_Δ.re-enc outputs the same ciphertext as above, except c_1 replaced by c'_1 and (t_1, t_1) appended to the list of evaluation epoch tuples (because we have started a new evaluation interval).

Finally, to decapsulate ciphertext $(c_1, vk, c_2, \sigma, ((t_{0,0}, t_{0,1}), \ldots, (t_{l-1,0}, t_{l-1,1}), (t_{l,0}, t_{l,1})))$, for each tuple $(t_{i,0}, t_{i,0})$ for i from l to 0, sequentially for t' from $t_{i,1}$ to $t_{i,0}$, the receiver first retrieves $(sk_{t'}, t'_0) \leftarrow SK[t']$, where t'_0 was the lower endpoint of the active interval *when epoch t' was created.* Then, IKMR_Δ.dec inverts c_1 using the FSP secret key $sk_{t'}$ with respect to the FSP instantiation's epoch $t_{0,0} - t'_0$ (the same epoch on which it was evaluated). For RCCA-security, the receiver also verifies $\text{OTS.ver}(vk, c_2, \sigma) = 1$ and recomputes c_1 and c_2 to check well-formedness. Then, using the final inverse $x \leftarrow c_1$ from above, IKMR.dec outputs $h(x)$ as the key.

Efficiency. Before formally analyzing the security of IKMR_Δ we will provide some bounds on the efficiency of the construction. It is clear that the size of every public key is proportional to $\Delta(t_1 - t_0)$. We will now attempt to bound the size of every (even re-encapsulated) ciphertext. First, the components needed for RCCA security, vk, c_2, σ only add $O(\lambda)$ bits and stay untouched even after several re-encapsulations. Now, for each re-encapsulation during epoch t_1, if the last epoch $t_{l,1}$ in which the ciphertext was (re-)encapsulated is such that $t_{l,1} + 1 \in [t_1 - \Delta(t_1 - t_0) - 1, t_1]$ then PK contains FSP public keys $pk_{t'}$ for $t' \in [t_{l,1} + 1, t_1]$. Therefore, the c part of the ciphertext can be re-evaluated sequentially using the FSP public keys $pk_{t'}$ for all such t', and the last epoch $t_{l,1}$ of the last evaluation interval of the ciphertext can be replaced with t_1. Observe that in this case, the ciphertext *does not grow*, assuming that each epoch can be represented using some fixed $\beta = O(\log \lambda)$ number of bits. Indeed, only if $t_{l,1} + 1 \notin [t_1 - \Delta(t_1 - t_0) - 1, t_1]$ and therefore PK does not contain a FSP public key for $t_{l,1}$, must the ciphertext grow. In this case, IKMR_Δ.re-enc skips to evaluating the c part of the ciphertext on only pk_{t_1}, and appends to the ciphertext evaluation interval (t_1, t_1). Here, the ciphertext grows by $O(\beta)$ bits.

Consider the case that some ciphertext is (re-)encapsulated in epochs $t^1 < \cdots < t^r$ such that for each $i \in [r]$, the active interval when epoch t^i was created was $[t_0^i, t_1^i]$ (where $t_1^i = t^i$). Then, it is clear that if for any $i \in [r]$, $t_0^i > t^1$, the ciphertext size becomes 0, as the original creation epoch t^1 is outside of the active interval and thus the re-encryptor sets the ciphertext to $\perp$. Otherwise, let $\delta_i = \Delta(t_1^i - t_0^i)$ for $i \in [n]$ and let $\delta_1^*, \ldots, \delta_r^*$ be $\delta_1, \ldots, \delta_r$ sorted in ascending order. In Lemma 1 below, we in fact show that the number of times G the ciphertext grows is bounded by

$$G \leq \operatorname{argmax}_g \left\{ t^1 + \sum_{i=1}^{g} \delta_i^* \leq t^r \right\}. \tag{1}$$

Therefore, the ciphertext size is bounded by $O(\lambda + G \cdot \beta)$, where β is the number of bits used to represent each epoch.

Lemma 1. *Given (re-)encapsulation epochs $t^1 < \cdots < t^r$ with active intervals $[t_0^i, t_1^i]$ when created, let $\delta_i = \Delta(t_1^i - t_0^i)$ for $i \in [r]$. Let $\delta_1^*, \ldots, \delta_r^*$ be $\delta_1, \ldots, \delta_r$ sorted in ascending order. Then, the number of times G the ciphertext can grow is bounded by Eq. 1.*

Proof. The ciphertext is first encapsulated at time t^1 and last re-encapsulated at time t^r. Moreover, we know that the j-th re-encryption only grows the ciphertext if $t_1^j - \delta_j > t_1^{j-1}$, or $t_1^j - t_1^{j-1} > \delta_j$. Thus, it must be that for those j s.t. the above is true: $t^1 + \sum_j \delta_j \leq t^r$. Therefore, the maximum number G of such j above corresponds to $\operatorname{argmax}_g\{t^1 + \sum_{i=1}^{g} \delta_i^* \leq t^r\}$. $\square$

Corollary 1. *G is bounded by $(t^r - t^1)/\min_j \delta_j$.*

Corollary 2. *If $(t_1^1 - t_0^1) = \cdots = (t_1^r - t_0^r) = T_1 - T_0$, then the number of times G the ciphertext can grow is bounded by $(t^r - t^1)/\Delta(T_1 - T_0)$.*

Security. Now, we show that construction IKMR$_\Delta$ of Fig. 3 is correct and secure. Before formally stating the theorem, we give some intuition on the security of the scheme. First, recall that the ABO secret key is not stored by the receiver. Now, let t^* be the epoch in which the challenge ciphertext is originally created and let t_{pub} be the epoch in which the (re-)encapsulated challenge ciphertext is made public. By the definition of the security game, it cannot be the case that any st_{t_0,t_1} with $t_0 \leq t^* \leq t_{pub} \leq t_1$ is ever leaked to the adversary. Indeed, it must be that either $t_1 < t_{pub}$ or $t^* < t_0$. In the former case, it is easy to see that *no* information about the FSP secret key $sk_{t_{pub}}$ generated for epoch t_{pub} is leaked to the adversary (beyond $pk_{t_{pub}}$). In the latter case, a version of $sk_{t_{pub}}$ may be leaked to the adversary, but only a version that has been updated past (its internal FSP epoch for corresponding IKMR$_\Delta$) epoch t^*. Therefore, by the security of FSP and ABO, the inverse of the final evaluations c_1, c_2 in the challenge ciphertext can be many possible values, and thus the same can be said about the original random $x \leftarrow_{\$} \{0,1\}^n$ sampled for the challenge ciphertext. As a result, from x, a key that is indistinguishable from random is extracted using the pair-wise independent hash function h. Moreover, for the RCCA security proof, we are able to always decrypt ciphertexts involving honest (re-)encapsulations using the public key output in epoch t_{pub} by first switching the ABO to lossy mode on branch vk^* before switching the FS-TDP to lossy mode, where vk^* is the sampled verification key for the challenge ciphertext, and then using the ABO to decapsulate instead of the FS-TDP (in the hybrid worlds, the receiver keeps the ABO secret key). Indeed, due to the unforgeability of the OTS scheme, the ABO will never have to invert on branch vk^*, and thus this modified decapsulation will always succeed. The proofs of the following Theorems are provided in the full version [8].

Theorem 2. *Let* FSP *be a family of correct, L-lossy, and $(T, \varepsilon_{\mathrm{FSP}})$-secure trapdoor permutations on common domain $\mathcal{X} = \{0,1\}^n$; let $\mathcal{H}_2 : \{0,1\}^n \to \{0,1\}^\ell$ be a family of pairwise independent hash functions where $\ell \leq k - 2\log(1/\varepsilon_{\mathcal{H}})$, for some $k = \omega(\log n)$ and some negligible $\varepsilon_{\mathcal{H}_2} = \mathsf{negl}(\lambda)$, where $\log(L) + \log(L') \geq n + k$;* OTS *be a strongly unforgeable one-time signature scheme where the verification keys are in $\{0,1\}^v$; and* ABO *be a family of correct, L'-lossy, and $(T, \varepsilon_{\mathrm{ABO}})$-secure all-but-one trapdoor functions on domain $\{0,1\}^n$. Then, for $T' \approx T$, the* IKMR *construction of Fig. 3 is correct and $((T', T^2 \cdot (O(1/2^\lambda) + \varepsilon_{\mathrm{OTS}} + 2 \cdot (\varepsilon_{\mathrm{ABO}} + \varepsilon_{\mathrm{FSP}}) + \varepsilon_{\mathcal{H}_2}))$-secure) in game* IND-RCCA$_{\mathrm{IKMR}}$ *of Fig. 2.*

Note that given the Lossy TDP of [4] with $L = 2^{n/4}$ and the ABO of [29] with $L' = 2^n/\lambda$, we indeed have that $\log(L) + \log(L') = 5n/4 - \log\lambda \geq n + \omega(\lg n)$.

5 IKEMR Construction from FS-PKE

While the advantage of our TDP-based IKEMR construction is the small ciphertext size, a disadvantage is the public key size. Using Forward-Secure Public-Key Encryption (FS-PKE), we can reduce the public key to an element of constant size at the cost of increasing the ciphertext size; furthermore, also the secret key size is slightly increased.

IKMR Construction. Since, for our TDP-based IKEMR construction, we already introduce FS-TDP as an abstraction layer, the primary difference towards our FS-PKE-based construction is the omission of parameter Δ. For security against active adversaries, we use an additional collision-resistant hash function. This simplifies the description of our construction from Fig. 4 significantly.

Initial key generation via IKMR.gen generates an FS-PKE key pair with one (update-)epoch for the FS-PKE secret key as well as a key for the hash function. Re-generating a key pair via IKMR.re-gen for interval $[t_0, t_1]$ generates an FS-PKE key pair with at most $t_1 - t_0 + 1$ update-epochs; the generated FS-PKE public key becomes the new IKEMR public key and the new FS-PKE secret

```
IKMR.gen($1^\lambda$):
  1. set $t_0, t_1 \leftarrow 1$, $SK[\cdot], PK \leftarrow \bot$
  2. generate $\kappa \leftarrow_\$ \{0,1\}^\lambda$
  3. generate $(sk, pk) \leftarrow_\$$
       FSE.gen$(1^\lambda, 1)$
  4. set $(SK[1], PK) \leftarrow$
       $((sk, t_0), pk)$
  5. return $((\kappa, SK, t_0, t_1),$
       $(\kappa, PK, t_0, t_1))$

IKMR.re-gen($(\kappa, SK, t_0, t_1)$)
  1. increment $t_1 \leftarrow t_1 + 1$
  2. generate $(sk, pk) \leftarrow_\$$
       FSE.gen$(1^\lambda, t_1 - t_0 + 1)$
  3. set $(SK[t_1], PK) \leftarrow$
       $((sk, t_0), pk)$
  4. return $((\kappa, SK, t_0, t_1),$
       $(\kappa, PK, t_0, t_1))$

IKMR.enc($(\kappa, PK, t_0, t_1)$):
  1. sample random $k \leftarrow_\$ \mathcal{K}$
  2. compute $c \leftarrow_\$$
       FSE.enc$(PK, t_1 - t_0, (k, \mathrm{H}_\kappa(t_1)))$
  3. return $(k, (c, (t_1)))$

IKMR.del($(\kappa, SK, t_0, t_1), \ell$):
  1. for $i \in [\ell]$:
     (a) set $SK[t_0 + i - 1] \leftarrow \bot$
     (b) for $t \in [t_0 + \ell, t_1]$:
           set $(sk, t') \leftarrow SK[t]$;
           update $sk' \leftarrow$ FSE.up$(sk)$;
           store $SK[t] \leftarrow (sk', t')$
  2. increment $t_0 \leftarrow t_0 + \ell$
  3. return $(\kappa, SK, t_0, t_1)$

IKMR.re-enc($(\kappa, PK, t_0, t_1), (c, (t_0^\circ, \ldots, t_l^\circ))$):
  1. if $t_0 > t_0^\circ$ return $\bot$
  2. compute $h \leftarrow \mathrm{H}_\kappa((t_0^\circ, \ldots, t_l^\circ, t_1))$
  3. compute
       $c \leftarrow_\$$ FSE.enc$(PK, t_0^\circ - t_0, (c, h))$
  4. return $(c, (t_0^\circ, \ldots, t_l^\circ, t_1))$

IKMR.dec($(\kappa, SK, t_0, t_1), (c, (t_0^\circ, \ldots, t_l^\circ))$):
  1. for $i$ from $l$ to 0:
     (a) let $(sk_i, t_0') \leftarrow SK[t_i^\circ]$; decrypt
           $(c, h) \leftarrow$ FSE.dec$(sk_i, t_0^\circ - t_0', c)$
     (b) if $h \neq \mathrm{H}_\kappa((t_0^\circ, \ldots, t_i^\circ))$: return $\bot$
  2. set $k \leftarrow c$
  3. return $k$
```

Fig. 4. FS-PKE-based IKEM with Re-Encapsulations construction. Text written in green is only needed for IND-RCCA$_{\mathrm{IKMR}}$ security. (Color figure online)

key is added to the decapsulation interval. Consequently, the IKEMR public key always only consists of a single FS-PKE public key. Deleting l slots from the decapsulation interval via IKMR.del removes the oldest l FS-PKE secret keys entirely; all remaining $t_1 - t_0 - l$ FS-PKE secret keys are updated l times each. This mechanism as well as the underlying rationale resemble those of our FS-TDP-based construction: only decapsulation of ciphertexts initially created after the beginning of the current decapsulation interval should be possible; in particular, ciphertexts encapsulated earlier but re-encapsulated later than that must not be recoverable.

At encapsulation via IKMR.enc, a randomly sampled key k is FS-PKE encrypted to the last update-epoch $t_1 - t_0$ of the current public key; the resulting ciphertext is appended with the current epoch index t_1. For re-encapsulation of ciphertext c with public key PK via IKMR.re-enc, c is FS-PKE encrypted to update-epoch $t_0^\circ - t_0$, where t_0° is the time at which the first version of c was initially created and t_0 was the oldest slot of the decapsulation interval

Initialization: Set $i^* \leftarrow -1$ and $D[\cdot] \leftarrow \perp$. Then initialize $(st_{sec}, st_{pub}) \leftarrow_\$ \mathrm{IRM.init}(1^\lambda)$ and output st_{pub}.

```
Write(i, ~C, d):
  1. execute $(st_{pub}, C) \leftarrow$
       $\mathrm{IRM.srvr\text{-}op}(st_{pub}, i, \mathrm{write})$
  2. $C \leftarrow \widetilde{C}$
  3. execute $(st_{sec}, C') \leftarrow_\$$
       $\mathrm{IRM.write}(st_{sec}, C, i, d)$
  4. execute $st_{pub} \leftarrow$
       $\mathrm{IRM.srvr\text{-}up}(st_{pub}, i, C')$
  5. if $C' \neq \perp$, set $D[i] \leftarrow d$ and if
       (also) $i = i^*$, set $i^* \leftarrow -1$
  6. return $C'$

Read(i, ~C):
  1. execute $(st_{pub}, C) \leftarrow$
       $\mathrm{IRM.srvr\text{-}op}(st_{pub}, i, \mathrm{read})$
  2. set att $\leftarrow 0$
  3. if $C \neq \widetilde{C}$, set $C \leftarrow \widetilde{C}$ and att $\leftarrow 1$
  4. execute $(st_{sec}, d) \leftarrow$
       $\mathrm{IRM.read}(st_{sec}, C, i)$
  5. if (att $= 0$ and $d \neq D[i]$) or
       att $= 1$ and $d \notin \{D[i], \perp\}$), **win**

Expose():
  1. return $\perp$ if $i^* \neq -1$
  2. return $st_{sec}$

Chall(i, ~C, d_0, d_1):
  1. execute $(st_{pub}, C) \leftarrow$
       $\mathrm{IRM.srvr\text{-}op}(st_{pub}, i, \mathrm{write})$
  2. $C \leftarrow \widetilde{C}$
  3. flip random coin $b \leftarrow_\$ \{0, 1\}$
  4. execute $(st_{sec}, C') \leftarrow_\$$
       $\mathrm{IRM.write}(st_{sec}, C, i, d_b)$
  5. execute $st_{pub} \leftarrow$
       $\mathrm{IRM.srvr\text{-}up}(st_{pub}, i, C')$
  6. if $C' \neq \perp$, set $D[i] \leftarrow d_b$ and $i^* \leftarrow i$
  7. disable **Chall** and return $C'$

Del(i, ~C):
  1. execute $(st_{pub}, C) \leftarrow$
       $\mathrm{IRM.srvr\text{-}op}(st_{pub}, i, \mathrm{del})$
  2. $C \leftarrow \widetilde{C}$
  3. execute $(st_{sec}, C') \leftarrow_\$$
       $\mathrm{IRM.del}(st_{sec}, C, i)$
  4. execute $st_{pub} \leftarrow$
       $\mathrm{IRM.srvr\text{-}up}(st_{pub}, i, C')$
  5. if $C' \neq \perp$, set $D[i] \leftarrow \perp$ and (also)
       if $i = i^*$ set $i^* \leftarrow -1$
  6. return $C'$
```

Fig. 5. IRAM correctness and security game. Text written in green is only needed for RINDIRM security. (Color figure online)

when PK was most recently re-generated. For active security, the a hash of the history of (re-)encapsulation epochs is added to each encrypted payload. Due to the FS-PKE re-encryption as well as the added hash value, the resulting IKEMR ciphertext grows with the number of IKEMR re-encapsulations by the encryption overhead of the FS-PKE scheme as well as the hash length. Additionally, each (re-)encapsulation attaches the current epoch index to the ciphertext, which is used at decapsulation via IKMR.dec to execute the matching FS-PKE decryptions; note this epoch list is only attached once to the outmost encryption layer.

Efficiency. Clearly, the public key size of this construction is constant for an FS-PKE scheme with constant sized public keys. For clarity, we use the original performance metric by Canetti et al. [17] that is based on the Gentry-Silverberg HIBE [24], where the public key is of constant size and the overhead of secret and public key is logarithmic in the total number of update-epochs, respectively. Based on this, our ciphertexts grow linearly in r, which is the number of re-encapsulations. This is increased by the ciphertext overhead of the underlying FS-PKE as well as additional epoch indexes. In total, this yields a ciphertext overhead of $O(r \cdot \log(t_1 - t_0))$. Finally, each secret key in the decapsulation interval is of size $\log(t_1 - t_0)$, which yields a total secret key size of $O((t_1 - t_0) \cdot \log(t_1 - t_0))$.

Security. Intuitively, almost the same argument works to prove security of the FS-PKE-based IKEMR construction as for our TDP-based one. For a challenge ciphertext c^* published at time t^*, none of the states exposed at time $t' < t^*$ contains the corresponding FS-PKE secret key. Furthermore, as soon as the initial creation time t_1 of the first version of c^* is deleted from the decapsulation interval, the FS-PKE secret key from time t^* was updated at least $t^* - t_1 + 1$ times. Based on this, instead of embedding a lossy TDP in epoch t^*, we replace the key, respectively ciphertext, that is FS-PKE (re-)encrypted at time t^* with a random string of the same length. Detecting this modification breaks the FS-PKE scheme. For security against active adversaries, we additionally bind the attached list of re-encapsulation epochs via the encrypted hash value. Intuitively, this yields the following Theorem that we formally prove in the full version [8].

Theorem 3. *Let* H *be a* $(T_{\mathrm{H}}, \varepsilon_{\mathrm{H}})$*-collision-resistant hash function and* FSE *be a correct and* $(T_{\mathrm{FSE}}, \varepsilon_{\mathrm{FSE}})$*-secure FS-PKE. Then the* IKMR *construction of Fig. 4 is correct and* $(T', \varepsilon_{\mathrm{H}} + T^2 \cdot \varepsilon_{\mathrm{FSE}})$*-secure in game* $\mathrm{IND\text{-}RCCA}_{\mathrm{IKMR}}$ *of Fig. 2, for* $T' \approx T_{\mathrm{H}} + T_{\mathrm{FSE}}$*.*

Alternative via FS-KEM and AEAD. In the full version [8], we propose an alternative construction based on FS-KEM and AEAD (instead of FS-PKE and collision-resistant hash functions). Without a formal proof, we claim that the security guarantees of both constructions are identical under suitable assumptions. The performance differences depend on the performance of the underlying building blocks. As mentioned above, ciphertexts of the FS-PKE construction in Fig. 4 grow based on the FS-PKE's encryption overhead o_{FSE} and the length of the hash l_{H}: $|c| = r * (o_{\mathrm{FSE}} + l_{\mathrm{H}}) + |k|$, where r is the number of re-encryptions. Ciphertexts of the FS-KEM construction given in the full version [8] grow based on the AEAD's encryption overhead o_{AE} and the FS-KEM's ciphertext length l_{FSK}: $|c| = r * (l_{\mathrm{FSK}} + o_{\mathrm{AE}}) + |k|$. The decryption time for the FS-PKE based construction is linear in the number of re-encryptions times the FS-PKE decryption time. If FS-KEM decryptions can be parallelized, the decryption time for the FS-KEM based construction is only linear in the number of re-encryptions times the AEAD decryption time. We also note that (without assuming Random Oracles) collision resistant hash functions can only be built from structured algebraic assumptions (see e.g., [12]), while AEAD can be built from symmetric primitives.

6 Minimizing Local State Size with External Storage

Our lower bound shows that the secret state st_{t_0,t_1} of any IKEM(R) scheme must be of size at least $t_1 - t_0$. Storing a secret state of such large size may be cumbersome for the receiver. Therefore, in this section, we introduce the IRAM primitive which the receiver may use to split st_{t_0,t_1} into a small secret component $st^{sec}_{t_0,t_1}$ (of size $O(1)$ in our construction) and public component $st^{pub}_{t_0,t_1}$ (still of size $O(t_1 - t_0)$ in our construction). Only the former needs to be securely stored by the receiver, while the latter can be stored by a server. IRAM allows for a generic interface to perform reads, writes, and deletions on the original state st_{t_0,t_1}, with minimal overhead. Any data that is stored in st_{t_0,t_1} should remain secure even if the adversary obtains several of the secret states before the data is written to st_{t_0,t_1} and after it is removed from st_{t_0,t_1}. We provide a definition that allows the server holding $st^{pub}_{t_0,t_1}$ to be actively corrupted (i.e., to deviate from the protocol arbitrarily), and also a definition in which the server is honest-but-curious). We call an IRAM that is secure even against an actively corrupted server, *robust.* The IRAM primitive can easily be composed with IKEM(R) to yield a secure IKEM(R) scheme with small secret state.

Interval RAM Definition. The IRAM primitive splits the storage of a dynamically-sized database into a secret component stored by the client and a public component stored by the server. For each read, write, and deletion operation on some virtual location i of the database, the server first executes a corresponding algorithm on the public state which outputs those cells C of the new public state that are relevant for the client-side operation. The server algorithm is deterministic based on the virtual location i of the database on which the client is operating.[3] The client then uses C received from the server to perform the operation and possibly uploads new cells C' with which the server should update the public state. In the case of an actively corrupted server, the adversary may send incorrect cells $\widetilde{C}$ to the client for operations, while for honest-but-curious server, the cells sent by the server are always the correct ones, C.

Syntax. An *Interval RAM* (IRAM) scheme IRM is a tuple of algorithms IRM = (IRM.init, IRM.srvr-op, IRM.read, IRM.write, IRM.del, IRM.srvr-up) with the following syntax:

- IRM.init$(1^\lambda) \rightarrow_\$ (st_{sec}, st_{pub})$ initializes the secret and public state of IRAM.
- IRM.srvr-op$(st_{pub}, i, op) \rightarrow (st_{pub}, C)$ the server executes operation $op \in \{\text{read}, \text{write}, \text{del}\}$ on virtual cell i of the database, updating st_{pub}, and returns those locations of st_{pub} that are needed by the client to execute op, via C.
- IRM.read$(st_{sec}, C, i) \rightarrow (st_{sec}, , d)$ using cells C of st_{pub} provided by the server, the client returns the i-th entry of the database, d.

[3] Deterministic server operations are common in outsourced database primitives, see e.g., [9,10,14,22].

- IRM.write$(st_{sec}, C, i, d) \rightarrow_{\$} (st_{sec}, C')$ using cells C of st_{pub} provided by the server, the client writes data d to the i-th entry of the database. In doing so, the client returns new public cells C' relevant to virtual cell i with which the server will replace the corresponding cells in st_{pub}.
- IRM.del$(st_{sec}, C, i) \rightarrow_{\$} (st_{sec}, C')$ using cells C provided by the server, the client deletes the i-th database entry. In doing so, the client returns new public cells C' relevant to virtual cell i with which the server will replace the corresponding cells in st_{pub}.
- IRM.srvr-up$(st_{pub}, C) \rightarrow st_{pub}$ the server places cells C in st_{pub} (the location of these cells in st_{pub} are implicitly encoded in C).

Correctness and Security. Now we define the correctness and security for an IRAM scheme. Intuitively, correctness dictates that when reading the i-th virtual location of the database, the data which was last written to the i-th virtual location must be returned. In the case of actively corrupted server, if the adversary does not honestly execute IRM.srvr-op() or IRM.srvr-up at some point, then the scheme can reject by outputting $\perp$. Security dictates that if any data d is written to cell i of the database, then it should remain private even if the adversary obtained several of the IRAM secret states *before* the write operation and obtains several secret states *after* data d of cell i is overwritten or deleted.

More formally, we define the correctness and security games XIND, $\mathrm{X} \in \{\mathrm{R}, \epsilon\}$ for IRAM in Fig. 5. The former is for $\underline{\mathrm{R}}$obust security, in which the adversary may arbitrarily deviate from the protocol specification, and the latter is for non-robust security, in which the adversary is assumed to be honest-but-curious. The green text in Fig. 5 is only for $\mathrm{RIND}_{\mathrm{IRM}}$. The game starts by initializing the IRAM via IRM.init, and outputting the public state st_{pub} to the adversary. It also stores variable $i^* \leftarrow -1$, which indicates if there is an active challenge, and if so the cell for which this challenge has been queried ($i^* = -1$ means there is no active challenge). For each **Write** query, the adversary specifies the virtual cell i and data d to write. The oracle first executes $(st_{pub}, C) \leftarrow$ IRM.srvr-op$(st_{pub}, i, \mathrm{write})$ to perform the server-side write operation and obtain the cells C necessary for the client-side operation; in the case of game $\mathrm{RIND}_{\mathrm{IRM}}$, the adversary actually inputs cells $\widetilde{C}$ to be used for the client-side operation (which is reflected by the oracle setting $C \leftarrow \widetilde{C}$). The oracle then executes IRM.write on input C, i, and d to obtain new secret state st_{sec} and cells C' which it uses to update st_{pub} via IRM.srvr-up. Then, the oracle stores data d in its own dictionary $D[i] \leftarrow d$ and checks if there is an active challenge for cell i ($i^* = i$), and if so resets $i^* \leftarrow -1$, since this query will overwrite the challenge data. This step is only applied in game $\mathrm{RIND}_{\mathrm{IRM}}$ if $C' \neq \perp$; i.e., only if the IRM.write operation was successful. Finally, the oracle returns the updated C' to the adversary. Oracle **Del** is specified in exactly the same way as **Write**, except instead of executing IRM.write, it executes IRM.del on virtual cell i.

For each query to **Chall**, the game acts similarly as to **Write**, except it sets challenge cell $i^* \leftarrow i$ (if $C' \neq \perp$ in the case of $\mathrm{RIND}_{\mathrm{IRM}}$; i.e., the IRM.write operation was successful), and flips a random coin b to determine on which of d_0 or d_1 IRM.write should be executed. Then, the game disables the oracle. Oracle **Read** takes as input from the adversary virtual cell i to read. The oracle first executes $(st_{pub}, C) \leftarrow \text{IRM.srvr-op}(st_{pub}, i, \text{read})$ to perform the server-side read operation and obtain the cells C necessary for the client-side operation; as with the other oracles, in the case of game RIND_{IRM}, the adversary actually inputs cells $\widetilde{C}$ to be used for the client-side operation. Also in the case of game RIND_{IRM}, if these adversarial cells do not match the cells from the honest IRM.srvr-op operation, $\widetilde{C} \neq C$, then the oracle sets $\text{att} \leftarrow 1$ to denote an active attack by the adversary. Note that the deterministic nature of IRM.srvr-op, IRM.srvr-up enables the game to detect active attacks in this way. Next, the oracle executes IRM.read on input i and cells C to receive data d. In the case of RIND_{IRM}, if $\text{att} = 0$, the oracle checks that d is equal to $D[i]$ which was last successfully written to cell i of the database; if $\text{att} = 1$, the oracle checks that either d is still equal to $D[i]$ or $d = \perp$ (indicating detection of the adversarial attack). For whichever value of att, if the check fails, the adversary wins the game, denoted by keyword **win** (in this case, the game just outputs the challenge bit b). In the non-robust game IND_{IRM}, the oracle only checks that $d = D[i]$. Finally, **Expose** simply returns the current secret state st_{sec} to the adversary, only if $i^* = -1$, i.e., only if there is not an active challenge.

Definition 14. *For* $\text{X} \in \{\epsilon, \text{R}\}$,[4] *an IRAM scheme* IRM *is* $(T, \varepsilon^{\text{xind}}_{\text{IRM}})$-secure *if for all adversaries* $\mathcal{A}$ *playing the security game* XIND_{IRM} *and running in time* T*:* $\Pr[b \leftarrow_{\$} \text{XIND}_{\text{IRM}}(\mathcal{A})] \leq 1/2 + \varepsilon^{\text{xind}}_{\text{IRM}}$.

Due to space limitations, we defer the presentation of our tree-based IRM construction that only uses symmetric-key cryptography to the full version [8]. Intuitively, it combines the concepts of encrypted RAM constructions for which so far only FS guarantees were proven (e.g., [9,10,16]), with Memory Checker constructions (e.g., [10,14,22]). For n stored entries, our construction has secret state size of $O(1)$ and read/write overhead of $O(\log n)$, which matches the lower bound of [10] for FS encrypted RAM.

References

1. Alwen, J., Auerbach, B., Baig, M.A., Noval, M.C., Klein, K., Pascual-Perez, G., Pietrzak, K., Walter, M.: Grafting key trees: Efficient key management for overlapping groups. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part III. LNCS, vol. 13044, pp. 222–253. Springer, Heidelberg (Nov 2021). https://doi.org/10.1007/978-3-030-90456-2_8
2. Alwen, J., Coretti, S., Dodis, Y., Tselekounis, Y.: Security analysis and improvements for the IETF MLS standard for group messaging. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 248–277. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56784-2_9
3. Alwen, J., Coretti, S., Jost, D., Mularczyk, M.: Continuous group key agreement with active security. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 261–290. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/978-3-030-64378-2_10

[4] ϵ indicates the ‘empty string’.

4. Auerbach, B., Kiltz, E., Poettering, B., Schoenen, S.: Lossy trapdoor permutations with improved lossiness. In: Matsui, M. (ed.) CT-RSA 2019. LNCS, vol. 11405, pp. 230–250. Springer, Heidelberg (Mar 2019). https://doi.org/10.1007/978-3-030-12612-4_12
5. Balli, F., Rösler, P., Vaudenay, S.: Determining the core primitive for optimally secure ratcheting. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 621–650. Springer, Heidelberg (Dec 2020). https://doi.org/10.1007/978-3-030-64840-4_21
6. Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E., Cohn-Gordon, K.: The Messaging Layer Security (MLS) Protocol. RFC 9420 (Jul 2023). https://doi.org/10.17487/RFC9420, https://www.rfc-editor.org/info/rfc9420
7. Bienstock, A., Dodis, Y., Rösler, P.: On the price of concurrency in group ratcheting protocols. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 198–228. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/978-3-030-64378-2_8
8. Bienstock, A., Dodis, Y., Rösler, P., Wichs, D.: Interval key-encapsulation mechanism. IACR Cryptol. ePrint Arch. (2024), https://eprint.iacr.org/2024/1454, full version of this article
9. Bienstock, A., Dodis, Y., Tang, Y.: Multicast key agreement, revisited. In: Galbraith, S.D. (ed.) CT-RSA 2022. LNCS, vol. 13161, pp. 1–25. Springer, Heidelberg (Mar 2022).https://doi.org/10.1007/978-3-030-95312-6_1
10. Bienstock, A., Dodis, Y., Yeo, K.: Forward secret encrypted RAM: Lower bounds and applications. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part III. LNCS, vol. 13044, pp. 62–93. Springer, Heidelberg (Nov 2021). https://doi.org/10.1007/978-3-030-90456-2_3
11. Bienstock, A., Rösler, P., Tang, Y.: Asmesh: Anonymous and secure messaging in mesh networks using stronger, anonymous double ratchet. In: CCS '23: 2023 ACM SIGSAC Conference on Computer and Communications Security 2023. ACM (2023)
12. Bitansky, N., Degwekar, A.: On the complexity of collision resistant hash functions: New and old black-box separations. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part I. LNCS, vol. 11891, pp. 422–450. Springer, Heidelberg (Dec 2019).https://doi.org/10.1007/978-3-030-36030-6_17
13. Blaze, M., Bleumer, G., Strauss, M.: Divertible protocols and atomic proxy cryptography. In: Nyberg, K. (ed.) EUROCRYPT'98. LNCS, vol. 1403, pp. 127–144. Springer, Heidelberg (May / Jun 1998).https://doi.org/10.1007/BFb0054122
14. Blum, M., Evans, W.S., Gemmell, P., Kannan, S., Naor, M.: Checking the correctness of memories. In: 32nd FOCS. pp. 90–99. IEEE Computer Society Press (Oct 1991). https://doi.org/10.1109/SFCS.1991.185352
15. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40041-4_23
16. Boneh, D., Lipton, R.J.: A revocable backup system. In: USENIX Security Symposium. pp. 91–96 (1996)
17. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (May 2003).https://doi.org/10.1007/3-540-39200-9_16
18. Canetti, R., Krawczyk, H., Nielsen, J.B.: Relaxing chosen-ciphertext security. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 565–582. Springer, Heidelberg (Aug 2003).https://doi.org/10.1007/978-3-540-45146-4_33

19. Choi, G., Durak, F.B., Vaudenay, S.: Post-Compromise Security in Self-Encryption. In: Tessaro, S. (ed.) 2nd Conference on Information-Theoretic Cryptography (ITC 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 199, pp. 25:1–25:23. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). https://doi.org/10.4230/LIPIcs.ITC.2021.25, https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITC.2021.25
20. Davidson, A., Deo, A., Lee, E., Martin, K.: Strong post-compromise secure proxy re-encryption. In: Jang-Jaccard, J., Guo, F. (eds.) ACISP 19. LNCS, vol. 11547, pp. 58–77. Springer, Heidelberg (Jul 2019). https://doi.org/10.1007/978-3-030-21548-4_4
21. Dodis, Y., Karthikeyan, H., Wichs, D.: Updatable public key encryption in the standard model. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part III. LNCS, vol. 13044, pp. 254–285. Springer, Heidelberg (Nov 2021). https://doi.org/10.1007/978-3-030-90456-2_9
22. Dwork, C., Naor, M., Rothblum, G.N., Vaikuntanathan, V.: How efficient can memory checking be? In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 503–520. Springer, Heidelberg (Mar 2009).https://doi.org/10.1007/978-3-642-00457-5_30
23. Fuchsbauer, G., Kamath, C., Klein, K., Pietrzak, K.: Adaptively secure proxy re-encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 317–346. Springer, Heidelberg (Apr 2019). https://doi.org/10.1007/978-3-030-17259-6_11
24. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (Dec 2002).https://doi.org/10.1007/3-540-36178-2_34
25. Haidar, C.A., Passelègue, A., Stehlé, D.: Efficient updatable public-key encryption from lattices. 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 V. Lecture Notes in Computer Science, vol. 14442, pp. 342–373. Springer (2023)
26. Jaeger, J., Stepanovs, I.: Optimal channel security against fine-grained state compromise: The safety of messaging. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 33–62. Springer, Heidelberg (Aug 2018).https://doi.org/10.1007/978-3-319-96884-1_2
27. Jost, D., Maurer, U., Mularczyk, M.: Efficient ratcheting: Almost-optimal guarantees for secure messaging. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 159–188. Springer, Heidelberg (May 2019).https://doi.org/10.1007/978-3-030-17653-2_6
28. Miao, P., Patranabis, S., Watson, G.J.: Unidirectional updatable encryption and proxy re-encryption from DDH. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part II. LNCS, vol. 13941, pp. 368–398. Springer, Heidelberg (May 2023).https://doi.org/10.1007/978-3-031-31371-4_13
29. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 187–196. ACM Press (May 2008).https://doi.org/10.1145/1374376.1374406
30. Perrin, T., Marlinspike, M.: The double ratchet algorithm (2016), https://signal.org/docs/specifications/doubleratchet/
31. Poettering, B., Rösler, P.: Asynchronous ratcheted key exchange. Cryptology ePrint Archive, Report 2018/296 (2018), https://eprint.iacr.org/2018/296

32. Poettering, B., Rösler, P.: Towards bidirectional ratcheted key exchange. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 3–32. Springer, Heidelberg (Aug 2018). https://doi.org/10.1007/978-3-319-96884-1_1
33. Rösler, P., Mainka, C., Schwenk, J.: More is less: On the end-to-end security of group chats in signal, whatsapp, and threema. In: 2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018. pp. 415–429. IEEE (2018)
34. Rösler, P., Slamanig, D., Striecks, C.: Unique-path identity based encryption with applications to strongly secure messaging. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 3–34. Springer, Heidelberg (Apr 2023).https://doi.org/10.1007/978-3-031-30589-4_1

Pairing-based Cryptography

Tightly Secure Non-interactive BLS Multi-signatures

Renas Bacho[1,2](✉) and Benedikt Wagner[3]

[1] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
renas.bacho@cispa.de
[2] Saarland University, Saarbrücken, Germany
[3] Ethereum Foundation, Berlin, Germany
benedikt.wagner@ethereum.org

Abstract. Due to their simplicity, compactness, and algebraic structure, BLS signatures are among the most widely used signatures in practice. For example, used as multi-signatures, they are integral in Ethereum's proof-of-stake consensus. From the perspective of concrete security, however, BLS (multi-)signatures suffer from a security loss linear in the number of signing queries. It is well-known that this loss can not be avoided using current proof techniques.

In this paper, we introduce a new variant of BLS multi-signatures that achieves tight security while remaining fully compatible with regular BLS. In particular, our signatures can be seamlessly combined with regular BLS signatures, resulting in regular BLS signatures. Moreover, it can easily be implemented using existing BLS implementations in a black-box way. Our scheme is also one of the most efficient non-interactive multi-signatures, and in particular more efficient than previous tightly secure schemes. We demonstrate the practical applicability of our scheme by showing how proof-of-stake protocols that currently use BLS can adopt our variant for fully compatible opt-in tight security.

Keywords: Non-Interactive · Multi-Signatures · BLS Signatures · Tightness · Pairings

1 Introduction

One of the most widely used digital signature schemes is due to Boneh, Lynn, and Shacham (BLS) [17]. BLS signatures play a crucial role in many decentralized applications such as Chia [19,33], randomness beacons [2,46], lotteries [11,30], and Ethereum's proof-of-stake (PoS) consensus [26]. They are not only simple and efficient, but they also possess several attractive algebraic properties. A particularly useful property is their support for efficient *non-interactive multi-signatures* [12,14]. For instance, suppose Alice, Bob, and Charlie each have a BLS secret key and use it to sign a message m individually. These individual signatures can be aggregated into a single BLS signature for m, which can be

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15485, pp. 397–422, 2025.
https://doi.org/10.1007/978-981-96-0888-1_13

verified against a combination of their public keys. Informally, this aggregated signature certifies that all three parties have signed m. This multi-signature feature is central to Ethereum's PoS mechanism, and it is also the subject of this work. In particular, we focus on the concrete security of BLS multi-signatures, as explained next.

Concrete Security of BLS. The security proof for BLS (multi-signatures) follows a straightforward reduction approach: assuming an efficient adversary $\mathcal{A}$ breaking BLS, one can construct an efficient reduction $\mathcal{R}$ that breaks the Computational Diffie-Hellman (CDH) assumption. Specifically, if $\mathcal{A}$ breaks BLS with probability ϵ, then $\mathcal{R}$ breaks CDH with probability at least $\epsilon' = \epsilon/\Theta(q_s)$, where q_s denotes the number of signatures that $\mathcal{A}$ learns. For practical values of q_s, this results in a relatively *loose* security bound: if $q_s = 2^{30}$ and CDH is 128-bit hard, this proof only guarantees 98 bits of security for BLS.

Tightness and Impossibility. It would be highly desirable to have a *tight* security proof, meaning a proof where $\epsilon' \approx \epsilon$. Unfortunately, such a tight proof is not possible for BLS multi-signatures. This is because a tight proof for BLS multi-signatures would imply a tight proof for single-signer BLS signatures, and existing impossibility results rule this out for unique signatures like BLS [4,21,35]. In contrast, a certain non-unique variant of BLS signatures can be proven tightly secure [31,36]. However, this variant sacrifices many of the desirable algebraic properties of the original BLS scheme.

Our Goal. While BLS cannot achieve tight security, we can still explore the following question:

Is there a tightly secure and non-interactive multi-signature compatible with standard BLS signatures?

Here, we should clarify what we mean by *compatibility*. Clearly, it cannot mean that signature verification is exactly the same as in BLS, due to the aforementioned impossibility results [4,21,35]. Instead, the minimal requirement for compatibility should be: (1) verification and signing algorithms can easily be obtained by invoking BLS signing and verification in a black-box manner, and (2) signers using the new scheme should be able to combine their signatures with those of legacy signers using plain BLS, without requiring significant changes in the verification process.

1.1 Our Contribution

We affirmatively address this question by constructing a variant of BLS multi-signatures that is efficient, tightly secure, and compatible with standard BLS signatures, as outlined next.

Security. Our scheme achieves tight security based on the CDH assumption in the random oracle model (ROM). In particular, we do not rely on the algebraic group model (AGM) [27] or the knowledge of secret key model (KOSK) [12]. We compare the security guarantees of our scheme with existing non-interactive multi-signatures in Table 1.

Efficiency. We compare the efficiency of non-interactive multi-signatures in Table 2. Our scheme is (almost) as efficient as regular BLS multi-signatures: signing involves computing one hash followed by calling regular BLS signing, while verification and aggregation maintain the same efficiency as regular BLS. Notably, our scheme outperforms the previous tightly secure schemes, BNN07 [8] and QLH12 [49], in terms of efficiency.

Compatibility and Applications. The core of our result is a new tightly secure signature scheme. Intuitively, a signer randomly uses one of two BLS keys for each message. Consequently, our signatures can be aggregated with regular BLS signatures, resulting in a standard BLS signature[1]. Our scheme can be implemented on top of existing BLS implementations in a black-box manner. As an application, we consider proof-of-stake (PoS) blockchains utilizing BLS, such as Ethereum [26]. In this context, our results demonstrate how to operate a validator with tight security while remaining compatible with existing validators. For further details, we refer to Sect. 5.

Remark 1 (Tightness and Compatibility). We argue that compatibility issues are often a reason why many tight schemes are not even being considered for deployment in practice. As an example, note that a long line of research has focused on Schnorr-compatible multi-party signatures, e.g. [1,22,43] and references therein. Such schemes are currently being implemented in Bitcoin and even about to be standardized. On the contrary, tightly secure variants which are not compatible (e.g., [47,48]) are not even considered for deployment and purely academic. Our scheme stands out as being compatible and tightly secure at the same time, which means that it is much more likely that this will find applications in practice.

1.2 More on Related Work

In this section, we discuss related work in more detail. Especially, we discuss previous results on non-interactive multi-signatures in general, and results specifically on BLS multi-signatures.

Multi-signatures. Multi-signatures have been introduced by Itakura and Nakamura [34] and later formalized in the *plain public key model* by Bellare and Neven [9]. In this model, each signer independently generates its own public-secret key pair. A major concern in this setting are rogue-key attacks, in which the adversary would choose its public key as a function of an honest user's key, allowing him to create forgeries easily. Such attacks have hindered progress in early stages [37,38,41,42,45]. In order to prevent rogue-key attacks, Boldyreva [12] has introduced the *knowledge of secret key (KOSK) model* for multi-signature schemes which was adopted by many subsequent works [22,24,39]. In this model, it is assumed that the adversary must reveal its secret keys at public key registration directly. For a discussion on this model with its drawbacks, we refer to [9,51]. Many multi-signature schemes with several rounds of communication per signature have been proposed. Three-round multi-signature

[1] The signature is not unique because it is valid for one of multiple possible keys.

Table 1. Comparison of non-interactive multi-signature schemes in the pairing setting. We compare under which hardness assumption the scheme is proven secure, the asymptotic tightness loss of the security proof, and under which idealized model the scheme is proven secure. Here, we do not consider proofs in the algebraic group model (AGM). We denote the number of random oracle and signing queries by q_h and q_s, respectively, and the advantage of an adversary against the scheme by ϵ. For LOSSW06 [39], ℓ denotes the bit-length of messages. Further, wBDHI denotes the weak bilinear Diffie-Hellman inversion assumption [13], ROM denotes the random oracle model, and KOSK denotes the knowledge of secret key model [12].

Scheme	Assumption	Loss	Idealization
BLS [12]	CDH	$\Theta(q_s)$	ROM
RY07 [51]	CDH	$\Theta(q_s)$	ROM
BDN18 [14]	CDH	$\Theta(q_h^2/\epsilon)$	ROM
LOSSW06 [39]	CDH	$\Theta(\ell q_s)$	KOSK
QX10 [50]	CDH	$\Theta(q_s^2 q_h/\epsilon)$	ROM
DGNW20 [25]	wBDHI	$\Theta(q_h)$	ROM
BNN07 [8]	CDH	$\Theta(1)$	ROM
QLH12 [49]	CDH	$\Theta(1)$	ROM
BLSMS$_2$	CDH	$\Theta(1)$	ROM

Table 2. Comparison of non-interactive multi-signature schemes in the pairing setting. We assume that all constructions are instantiated with a symmetric pairing $e\colon \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$ and compare the size of a public key, signature share, the size of the signature, the computational cost per signer, and the computational cost for verification. We denote the size of a group element by $\langle\mathbb{G}\rangle$ (respectively $\langle\mathbb{G}_T\rangle$), the number of signers by N, and the number of exponentiations, pairings, and k-multi-exponentiations for $k \in \mathbb{N}$ by ex, pr, and ex^k, respectively. For LOSSW06 [39], ℓ denotes the bit-length of messages.

Scheme	Public Key	Sig Share	Signature	Cost (Sig)	Cost (Ver)
BLS [12]	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	1ex	2pr
RY07 [51]	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	1ex	2pr
BDN18 [14]	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	1ex	2pr
LOSSW06 [39]	$1\langle\mathbb{G}_T\rangle$	$2\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle$	2ex + 1ex$^\ell$	2pr + 1ex$^\ell$
QX10 [50]	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle$	1ex	2pr + 1ex^{N+1}
DGNW20 [25]	$1\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle$	4ex	3pr + 1ex
BNN07 [8]	$1\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle + 1$	$1\langle\mathbb{G}\rangle + N$	1ex	$(N+1)$pr
QLH12 [49]	$1\langle\mathbb{G}\rangle$	$2\langle\mathbb{G}\rangle + 1$	$4\langle\mathbb{G}\rangle$	2ex	4pr
BLSMS$_2$	$2\langle\mathbb{G}\rangle$	$1\langle\mathbb{G}\rangle + 1$	$1\langle\mathbb{G}\rangle + N$	1ex	2pr

schemes have been constructed in [5,9,14,28,40,41], all of which base their security on standard assumptions, specifically the Decisional Diffie-Hellman (DDH) assumption and the Discrete Logarithm (DLOG) assumption. Further, two-round multi-signature schemes have been constructed [1,7,18,22,23,43,44,47,48,52], some of which are partially non-interactive (i.e., the first signing round is message-independent and can be preprocessed) [43,52], while others achieve tight security [47,48]. In this work, we focus specifically on *non-interactive* multi-signatures.

Non-interactive Multi-signatures. A non-interactive multi-signature scheme is a multi-signature scheme which requires only a single round of communication among a set of n parties to produce a signature. Namely, each party outputs a signature share, and then the n signature shares can be (publicly) combined into a single short signature. Despite its practical relevance, there are only a few non-interactive multi-signatures in the literature, which we briefly discuss next. The first non-interactive multi-signature scheme is the BLS multi-signature proposed by Boldyreva [12]. As for single-signer BLS, its security is based on the Computational Diffie-Hellman (CDH) assumption and has a security loss of $\Theta(q_s)$ where q_s denotes an upper bound on the number of allowed signing queries. Further, the security proof relies on the KOSK model. Several follow-up works [6,14,51] have proposed variants of the BLS multi-signature, eliminating the KOSK assumption. We will later elaborate in more detail on these schemes. Subsequently, several other schemes have been proposed [8,39]. The scheme proposed by Lu et al. [39] is based on the Waters signature scheme [53]. Its security is based on the CDH assumption and it has a security loss of $\Theta(\ell q_s)$ where ℓ denotes the bit-length of messages. The security proof relies on the KOSK model. The scheme proposed by Bellare et al. [8] is based on the aggregate signature scheme of Boneh et al. [15]. Here, a user i's signature σ_i is a key-prefixed BLS signature $\mathsf{H}(\mathsf{pk}_i, \mathsf{m})^{\mathsf{sk}_i}$ with the multi-signature being simply the product of individual signatures. Its security is based on the CDH assumption and comes with a security loss of $\Theta(q_s)$. Further, by using the Katz-Wang technique [31], the authors obtain a tight multi-signature. Notably, their schemes do not rely on the KOSK model. However, the final signatures require $n+1$ pairing evaluations for verification. Later, Qian and Xu [50] have proposed a multi-signature scheme that only requires two pairing evaluations (and one multi-exponentiation) for verification. Its security is based on the CDH assumption and it has a large security loss of $\Theta(q_s^2 q_h/\epsilon)$ where q_h denotes an upper bound on the number of allowed hash queries. A follow-up work by Qian et al. [49] improves upon this by proposing the first non-interactive multi-signature scheme with tight security and efficient verification. Their scheme is based on the Waters signature scheme and uses the Katz-Wang technique to obtain a tight security reduction from the CDH assumption. Notably, their scheme does not rely on the KOSK model. Finally, Drijvers et al. [25] have proposed a multi-signature scheme based on the Boneh-Boyen-Goh hierarchical identity-based encryption (HIBE) scheme [13]. Its security is based on the weak bilinear Diffie-Hellman inversion (wBDHI) assumption [13] for type-3 pairings and has a security loss of $\Theta(q_h)$. Notably, their scheme does not rely on the KOSK model.

BLS Multi-signatures. As stated above, the proof of security for the original BLS multi-signature by Boldyreva [12] relies on the KOSK model. This was improved upon by the scheme of Ristenpart and Yilek [51] leveraging proofs of possession (POPs) of secret keys to prevent rogue-key attacks without relying on the KOSK model. Still, the security is based on the CDH assumption and has a security loss of $\Theta(q_s)$. Later, Boneh et al. [14] have proposed another variant of the BLS multi-signature without relying on the KOSK model. Essentially, this is achieved by rerandomization of public keys of users as $\mathsf{pk}_i' \leftarrow \mathsf{pk}_i^{a_i}$ where $a_i := \mathsf{H}_{\text{rand}}(\mathsf{pk}_i, \{\mathsf{pk}_1, \ldots, \mathsf{pk}_N\})$ for a random oracle H_{rand}. Their security proof is still based on the CDH assumption, but now additionally relies on rewinding which results in a very loose bound of $\Theta(q_h^2/\epsilon)$. More recently, Baldimtsi et al. [6] gave a tight security reduction for the BLS multi-signature with rerandomization of public keys based on the DLOG assumption. However, their security proof relies on the algebraic group model (AGM) [27]. The recent Internet Engineering Task Force (IETF) draft [16] specifies BLS signatures with proofs of possession for use in practical deployments. In fact, this is how BLS signatures on the Ethereum blockchain are used [26]. As such, none of the proposed variants for BLS multi-signatures has a tight security proof without relying on the AGM.

1.3 Paper Organization

In Sect. 2, we give an informal but detailed technical overview of our constructions and proof techniques. In Sect. 3, we formally recall relevant cryptographic background and definitions. Then, in Sect. 4, we formally present our construction and its analysis. We conclude in Sect. 5, where we discuss an application to proof-of-stake blockchains.

2 Technical Overview

In this section, we give an informal overview of our constructions and our proof techniques. We do so in two steps: first, we introduce a new tightly secure variant of standard BLS signatures. While there is already a tightly secure variant of BLS [36], our variant is structurally more compatible with standard BLS as we will see. In the second step, we then show how to lift this construction to the multi-signature setting while preserving tight security.

2.1 Tightly Secure and Structured BLS Signatures

Let us first review BLS signatures and existing ways to construct tightly secure variants thereof. For simplicity, most of this overview will be written assuming a symmetric pairing $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}$ is a cyclic group of prime order p with generator g.

BLS Signatures. A regular BLS signature for a message m with respect to public key $\mathsf{pk} = g^{\mathsf{sk}}$ is given as $\sigma = \mathsf{H}(\mathsf{m})^{\mathsf{sk}}$, where $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}$ is a random oracle. It will be instructive to review the non-tight security proof of BLS [17]: the goal is to give a reduction from the CDH assumption. This reduction gets as input two group elements $X = g^x$ and $Y = g^y$, and its goal is to compute g^{xy}. To this end, the reduction simulates the $\mathsf{EUF\text{-}CMA}$ security game with public key $\mathsf{pk} = X$ for the adversary. While doing so, it splits the message space into two partitions: (1) for most messages m, it will program $\mathsf{H}(\mathsf{m}) := g^{\gamma_\mathsf{m}}$, where $\gamma_\mathsf{m} \in \mathbb{Z}_p$ is a random exponent known to the reduction. Note that for these messages, the reduction can efficiently provide signatures to the adversary by returning $\sigma = X^{\gamma_\mathsf{m}}$, i.e., γ_m serves as a trapdoor; (2) for all other messages, it will embed the challenge Y into the hash: $\mathsf{H}(\mathsf{m}) := Y \cdot g^{\gamma_\mathsf{m}}$. For these messages, the reduction can efficiently obtain a CDH solution from a valid signature output by the adversary. As long as the adversary only queries signatures for messages from the first partition, and forges for a message in the second partition, the reduction succeeds. Unfortunately, this partitioning leads to a security loss linear in the number of signing queries. Indeed, it is known that such a loss is inherent for unique signatures like BLS [4,21].

Random Bits for Tight Security. It is well-known that with a minimal change, BLS signatures can be made tightly secure: to sign a message m, a signer would pseudorandomly derive a bit $\beta_\mathsf{m} \in \{0,1\}$ from the message, and then compute the signature as $\sigma = \mathsf{H}(\mathsf{m}, \beta_\mathsf{m})^{\mathsf{sk}}$. This is often called the Katz-Wang technique [31], and it enables the following tight security proof: for each message m, the reduction programs $\mathsf{H}(\mathsf{m}, \beta_\mathsf{m}) := g^{\gamma_\mathsf{m}}$ and $\mathsf{H}(\mathsf{m}, 1-\beta_\mathsf{m}) := Y \cdot g^{\gamma'_\mathsf{m}}$. That is, the reduction embeds a trapdoor in one branch, and the challenge in the second branch for each message. Obviously, the reduction can now compute σ using the trapdoor. On the other hand, the bit β_{m^*} for the forgery message m^* remains pseudorandom for the adversary, and so with probability 1/2, the $(1-\beta_{\mathsf{m}^*})$-branch is used in the forgery, which ultimately allows the reduction to solve CDH. Observe that this proof does not partition the message space.

Algebraic Structure Lost. While the Katz-Wang technique gives an elegant way to obtain tight security, we pay a price: BLS signatures have many desirable algebraic features, which the random bit variant does not. For instance, assume we have two BLS signatures $\sigma_A = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_A}$ and $\sigma_B = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_B}$ under different keys for the same message m. Then, the product $\sigma_A \cdot \sigma_B = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_A + \mathsf{sk}_B}$ is a valid signature under the product of the keys $\mathsf{pk}_A \cdot \mathsf{pk}_B$. This observation underlies the design of BLS multi-signatures. Now, consider the same setting for the Katz-Wang variant: as each signer has to compute its bit pseudorandomly, the two signatures are likely of the form $\sigma_A = \mathsf{H}(\mathsf{m}, 0)^{\mathsf{sk}_A}$ and $\sigma_B = \mathsf{H}(\mathsf{m}, 1)^{\mathsf{sk}_B}$. When we multiply them, we do not get a BLS signature under the product of keys.

Towards a Solution. The above example shows that, in order to preserve the algebraic structure of BLS signatures, it is essential to ensure every signer uses the same basis $\mathsf{H}(\mathsf{m})$ for a given message m. Still, resorting to standard BLS can not lead to tight security [4,21], as already explained. To make progress towards a solution, let us assume that we still have a pseudorandom bit β_m, but use it differently. Concretely, say a signer now holds two BLS public keys $\mathsf{pk}_0 = g^{\mathsf{sk}_0}$

and $\mathsf{pk}_1 = g^{\mathsf{sk}_1}$. Then, we could define the signature to be $\sigma = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_{\beta_\mathsf{m}}}$. For now, it is not clear at all that this leads to a tight security proof, but we can already see that this is much more compatible with BLS than the Katz-Wang variant: suppose Alice uses this variant, but Bob still uses regular BLS. Now say we have their two signatures $\sigma_A = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_{A,\beta_\mathsf{m}}}$ and $\sigma_B = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_B}$. Then, the product $\sigma_A \cdot \sigma_B$ is a *regular BLS signature* for m with respect to the key $\mathsf{pk}_{A,\beta_\mathsf{m}} \cdot \mathsf{pk}_B$.

Proving Security. As the previous example shows, the variant above is structurally compatible with regular BLS signatures. The question is if this variant is also tightly secure. Indeed reusing the Katz-Wang proof technique does not work: we only have one branch for each message. This means that for each message m, we have to decide whether we would embed the challenge or a trapdoor, i.e., we have to partition the message space. Fortunately, it turns out that we can still get a tight security proof. Say our reduction gets as input the CDH challenge $X = g^x$ and $Y = g^y$. As in the proof for regular BLS signatures, we want to embed X in the key and Y in some of the random oracle outputs. Of course, embedding X in a fixed key, say in pk_0, is not a good idea. This is because an adversary could potentially always choose to use pk_1 in its forgery and the scheme degenerates to regular BLS. Instead, say we embed X randomly: we sample a bit $\hat{\beta} \overset{\$}{\leftarrow} \{0,1\}$ at random and define $\mathsf{pk}_{\hat{\beta}} := X$, and make sure that the reduction knows $\mathsf{sk}_{1-\hat{\beta}}$. Next, the reduction partitions the message space. This has to be done in a way that ensures that the reduction can always simulate signatures, namely:

- *Trapdoor Partition.* If $\beta_\mathsf{m} = \hat{\beta}$, i.e., the signature is $\sigma = \mathsf{H}(\mathsf{m})^x$, the reduction embeds a trapdoor into $\mathsf{H}(\mathsf{m})$.
- *Challenge Partition.* Otherwise, the signature is $\sigma = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_{1-\hat{\beta}}}$, and the reduction can embed Y into $\mathsf{H}(\mathsf{m})$ because it can always simulate such signatures using $\mathsf{sk}_{1-\hat{\beta}}$.

Now, consider the adversary's forgery (m^*, σ^*). We can argue that the bit b_{m^*} is hidden from the adversary, so with probability 1/2 over the choice of this bit, the message m^* is in the challenge partition. Similarly, with probability 1/2 over the choice of $\hat{\beta}$ the forgery is with respect to $\mathsf{pk}_{\hat{\beta}}$. This means that with probability 1/4, the forgery contains g^{xy}, which the reduction can use to solve CDH. In this way, we get a tight security proof.

2.2 Tightly Secure BLS Multi-signatures

Equipped with our tightly secure variant of BLS signatures, we now turn our focus to multi-signatures. We will first recall common techniques to securely turn BLS signatures into multi-signatures. Throughout, we consider the simplified setting of two parties, Alice and Bob, signing a message m. Alice will generally be assumed to be our honest party, whereas Bob is assumed to be adversarial.

BLS Multi-signatures. As we have mentioned above, we can combine the two BLS signatures $\sigma_A = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_A}$ of Alice and $\sigma_B = \mathsf{H}(\mathsf{m})^{\mathsf{sk}_B}$ of Bob into a single signature $\sigma = \sigma_A \cdot \sigma_B$ for the key $\mathsf{pk}_A \cdot \mathsf{pk}_B$. It is a well-established fact that without further modification, this naive BLS multi-signature is insecure due to so-called *rogue-key attacks*: Bob could choose its key as $\mathsf{pk}_B := \mathsf{pk}_A^{-1} \cdot g^\delta$, which allows him to create valid multi-signatures for public key $\mathsf{pk}_A \cdot \mathsf{pk}_B = g^\delta$ without talking to Alice. Prominently, there are two ways to solve this issue:

– *Random Linear Combinations.* Instead of defining the multi-signature as $\sigma_A \cdot \sigma_B$, we define it as $\sigma_A^{a_A} \cdot \sigma_B^{a_B}$ and the aggregate public key as $\mathsf{pk}_A^{a_A} \cdot \mathsf{pk}_B^{a_B}$, where $(a_A, a_B) \in \mathbb{Z}_p$ are random coefficients derived using a random oracle.
– *Proofs of Possession.* A public key is only considered valid if it comes with a proof of possession[2] of the secret key [51]. Concretely, this proof is usually implemented as $\mathsf{G}(\mathsf{pk}_B)^{\mathsf{sk}_B}$ (for Bob), i.e., as a BLS signature using a different random oracle.

The latter approach is often used in practice, e.g., in Ethereum [26], and it will also be our focus. Ristenpart and Yilek [51] have shown that this approach is provably secure for regular BLS. Interestingly, their proof does not add any additional security loss: the security loss for the multi-signature is the same as the one for regular BLS. Luckily, their proof technique also applies to our tightly secure variant, except for some complications in the asymmetric pairing setting. This holds even if Alice uses our variant, and Bob uses regular BLS. In the following, we review the challenge, the proof strategy by Ristenpart and Yilek [51], and explain the complications when using asymmetric pairings.

A Closer Look at the Proof. When we want to prove the security of the BLS multi-signature with proofs of possession, we have to simulate our honest signer Alice for the adversary, and we have to turn the adversary's forgery into a CDH solution. While the former task did not change compared to the case of standard signatures, the latter task is more challenging in the multi-signature setting. This is because a forgery σ^* is now valid for a message m^* and *some* combined key $\mathsf{pk}_A \cdot \mathsf{pk}_B$, where pk_B is made up by Bob. More concretely, say we have embedded our CDH challenge $(X, Y) = (g^x, g^y)$ in Alice's key, such that $\mathsf{sk}_A = x$, and say we have managed that $\mathsf{H}(\mathsf{m}^*)$ contains the challenge Y, i.e., $\mathsf{H}(\mathsf{m}^*) = Y \cdot g^{\gamma_{\mathsf{m}^*}}$ for some γ_{m^*} known to the reduction. For simplicity, assume $\gamma_{\mathsf{m}^*} = 0$ for now. Then, the forgery has the form

$$\sigma^* = \mathsf{H}(\mathsf{m}^*)^{\mathsf{sk}_A + \mathsf{sk}_B} = g^{xy} \cdot Y^{\mathsf{sk}_B}.$$

So, if the reduction wants to compute the CDH solution g^{xy} from σ^*, it has to compute $Y^{\mathsf{sk}_B} = g^{y\mathsf{sk}_B}$. But the reduction does not know y and $\mathsf{pk}_B = g^{\mathsf{sk}_B}$ is made up by the adversary!

The Adversary Solves Our Problem. As already observed by Ristenpart and Yilek [51], we can let the adversary solve our problem via proofs of possession: The reduction would embed Y into random oracle G as well. In simplified terms,

[2] Enforcing such a valid proof of possession can of course be modeled as just another check in the signature verification algorithm.

assume that $\mathsf{G}(\mathsf{pk}_B) = Y$. In this case, the proof of possession $\mathsf{G}(\mathsf{pk}_B)^{\mathsf{sk}_B}$ is exactly the desired term Y^{sk_B}, and the reduction can use it to compute g^{xy}.

Complications in the Asymmetric Setting. So far, we have simplified notation by using the symmetric pairing setting. Indeed, the technique by Ristenpart and Yilek [51] for regular BLS multi-signatures only works in the symmetric pairing setting and the type-2 pairing setting where there is an efficiently computable isomorphism $\psi\colon \mathbb{G}_2 \to \mathbb{G}_1$. Making the analysis work for an asymmetric pairing without such an isomorphism, also known as the type-3 setting[3], leads to subtle challenges, which we outline next. Namely, recall that above we have assumed $\gamma_{\mathsf{m}^*} = 0$. In general, this will of course not be the case and the forgery will have the form

$$\sigma^* = \mathsf{H}(\mathsf{m}^*)^{\mathsf{sk}_A+\mathsf{sk}_B} = (Y \cdot g^{\gamma_{\mathsf{m}^*}})^{\mathsf{sk}_A+\mathsf{sk}_B} = g^{xy} \cdot X^{\gamma_{\mathsf{m}^*}} \cdot Y^{\mathsf{sk}_B} \cdot g^{\gamma_{\mathsf{m}^*}\mathsf{sk}_B}.$$

We have already discussed how the reduction can eliminate the term Y^{sk_B} using the proofs of possession. It can of course also eliminate the term $X^{\gamma_{\mathsf{m}^*}}$ using knowledge of X and γ_{m^*}. In the symmetric pairing setting, the reduction can also compute and remove the final term $g^{\gamma_{\mathsf{m}^*}\mathsf{sk}_B} = \mathsf{pk}_B^{\gamma_{\mathsf{m}^*}}$. In the asymmetric setting, however, the reduction has the adversarially chosen keys pk_B only in one group, say $\mathbb{G}_2$, but the signature σ^* is in the other group $\mathbb{G}_1$, so the reduction needs to compute $g_1^{\gamma_{\mathsf{m}^*}\mathsf{sk}_B}$ over $\mathbb{G}_1$. It is not clear how to do that[4]. Our solution is to define the random oracles multiplicatively, namely, we set $\mathsf{H}(\mathsf{m}) := \mathsf{pk}_{1-b_{\mathsf{m}}}^{\gamma_{\mathsf{m}}}$ for each message m. It turns out that this enables a tight proof in the type-3 setting.

3 Preliminaries

Here, we define our notation and recall relevant cryptographic primitives and assumptions.

Notation. We denote the security parameter by λ and assume that all algorithms get 1^λ implicitly as input. We use standard cryptographic terminology like *negligible, overwhelming, PPT*. To sample an element x uniformly at random from a set W, we write $x \overset{\$}{\leftarrow} W$. We write $x \leftarrow \mathcal{D}$ if $\mathcal{D}$ is a distribution or a probabilistic algorithm. That is, writing $y \leftarrow \mathcal{A}(x)$ means that algorithm $\mathcal{A}$ is run with uniformly sampled random coins on input x and y is the output. If $\mathcal{A}$ is known to be deterministic, we write $y := \mathcal{A}(x)$ instead. We write $\mathbf{T}(\mathcal{A})$ to denote the running time of $\mathcal{A}$. We define $[N] := \{1, \ldots, N\} \subseteq \mathbb{N}$.

Computational Assumptions. Throughout this paper, we assume an algorithm $\mathsf{PGGen}(1^\lambda)$ that outputs cyclic groups $\mathbb{G}_1, \mathbb{G}_2$ of prime order p with generators $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, and a non-degenerate[5] pairing $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ into some target group $\mathbb{G}_T$.

[3] The type-3 setting is indeed the most common setting in practice.

[4] We could of course force Bob to additionally output its public key pk_B over $\mathbb{G}_1$, but this is generally not what happens in practice, so we refrain from doing so.

[5] Non-degenerate means that $\mathsf{e}g_1g_2$ is a generator of the group $\mathbb{G}_T$.

Definition 1 **(CDHAssumption).** *We say that the* CDH *assumption holds relative to* PGGen*, if for all PPT algorithms* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{A},\mathsf{PGGen}}(\lambda) := \Pr\left[z = xy \;\middle|\; \begin{array}{l}(\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda),\\ x, y \xleftarrow{\$} \mathbb{Z}_p,\ X_1 := g_1^x,\ X_2 := g_2^x,\ Y := g_1^y\\ g_1^z \leftarrow \mathcal{A}(\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e, X_1, Y, X_2)\end{array}\right]$$

Digital Signatures. We define the syntax of digital signatures and the standard notion of unforgeability under chosen-message attacks [32].

Definition 2 (Signature Scheme). *A signature scheme is a tuple of PPT algorithms* $\mathsf{SIG} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{par}$ *takes as input the security parameter* 1^λ *and outputs global system parameters* par*. We assume that* par *implicitly defines sets of public keys, secret keys, messages and signatures, respectively. All algorithms related to* SIG *take* par *at least implicitly as input.*
- $\mathsf{Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$ *takes as input system parameters* par*, and outputs a public key* pk *and a secret key* sk*.*
- $\mathsf{Sig}(\mathsf{sk}, \mathsf{m}) \to \sigma$ *takes as input a secret key* sk *and a message* m *and outputs a signature* σ*.*
- $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma) \to b$ *is deterministic, takes as input a public key* pk*, a message* m*, and a signature* σ*, and outputs a bit* $b \in \{0, 1\}$*.*

We require that SIG *is complete in the following sense. For all* $\mathsf{par} \in \mathsf{Setup}(1^\lambda)$*, all* $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{Gen}(\mathsf{par})$*, and all messages* m*, we have*

$$\Pr[\mathsf{Ver}(\mathsf{pk}, \mathsf{m}, \sigma) = 1 \mid \sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{m})] = 1.$$

Definition 3 **($\mathsf{EUF\text{-}CMA}$Security for Signatures).** *Let* $\mathsf{SIG} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Ver})$ *be a signature scheme. Consider an adversary* $\mathcal{A}$*. Further, consider the game* $\mathbf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{SIG}}(\lambda)$ *defined as follows:*

1. *Run* $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{par})$*.*
2. *Initialize* $\mathcal{Q} := \emptyset$ *and let* O *be the following oracle:*
   - $\mathrm{O}(\mathsf{m})$: *Take as input* m*, set* $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{m}\}$*, and return* $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, \mathsf{m})$*.*
3. *Run* $\mathcal{A}$ *on input* $(\mathsf{par}, \mathsf{pk})$ *and with access to oracle* O*. Obtain* (m^*, σ^*) *from* $\mathcal{A}$*.*
4. *Output* 1 *if and only if* $\mathsf{Ver}(\mathsf{pk}, \mathsf{m}^*, \sigma^*) = 1$ *and* $\mathsf{m}^* \notin \mathcal{Q}$*.*

We say that SIG *is* $\mathsf{EUF\text{-}CMA}$ *secure, if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{SIG}}(\lambda) := \Pr\left[\mathbf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{SIG}}(\lambda) \Rightarrow 1\right].$$

Next, we recall the signature scheme due to Boneh, Lynn, and Shacham [17]. It is well-known that it is (non-tightly) $\mathsf{EUF\text{-}CMA}$ secure based on the CDH assumption [17,20]. Throughout, we assume that signatures are in $\mathbb{G}_1$ and public keys are in $\mathbb{G}_2$. By symmetry, all of our results apply if the roles are reversed.

Definition 4 (BLS Signatures [17]). *Consider a random oracle* $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}_1$*. The signature scheme*[6] $\mathsf{BLS} = (\mathsf{BLS.Setup}, \mathsf{BLS.Gen}, \mathsf{BLS.Sig}^{\mathsf{H}}, \mathsf{BLS.Ver}^{\mathsf{H}})$ *is defined via the following algorithms:*

- $\mathsf{BLS.Setup}(1^\lambda) \to \mathsf{par}$*: Define* $\mathsf{par} := (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda)$*.*
- $\mathsf{BLS.Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$*: Sample* $\mathsf{sk} \overset{\$}{\leftarrow} \mathbb{Z}_p$ *and set* $\mathsf{pk} := g_2^{\mathsf{sk}}$*.*
- $\mathsf{BLS.Sig}^{\mathsf{H}}(\mathsf{sk}, \mathsf{m}) \to \sigma$*: Set* $\sigma := \mathsf{H}(\mathsf{m})^{\mathsf{sk}}$*.*
- $\mathsf{BLS.Ver}^{\mathsf{H}}(\mathsf{pk}, \mathsf{m}, \sigma) \to b$*: Return* $b = 1$ *if* $\mathsf{e}\mathsf{H}(\mathsf{m})\mathsf{pk} = \mathsf{e}\sigma g_2$*. Otherwise, return* $b = 0$*.*

Lemma 1. *If the* CDH *assumption holds relative to* PGGen *and* $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}_1$ *is modeled as a random oracle, then* BLS *is* $\mathsf{EUF\text{-}CMA}$ *secure. More precisely, for every PPT algorithm* $\mathcal{A}$ *that makes at most* Q *queries to* O*, there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ *and*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{BLS}}(\lambda) \leq \Theta(Q) \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).$$

Multi-signatures. Next, we give a definition for multi-signatures, specifically, non-interactive multi-signatures. In such a scheme, each signer independently generates its key pair via a key generation algorithm Gen. To sign a message, each signer locally uses its secret key to compute a signature via an algorithm Sig. A list of such signatures for the same message can then be combined into a single signature via an algorithm Comb. As a result, one obtains a signature that verifies for the given message and with respect to the list of public keys. Note that trivially, we obtain a multi-signature scheme by letting Comb concatenate the signatures. However, the goal should always be to construct non-trivial multi-signatures in a sense that Comb outputs a signature much smaller than the concatenation.

Definition 5 (Multi-signature Scheme). *A multi-signature scheme is a tuple of PPT algorithms* $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Comb}, \mathsf{Ver})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{par}$ *takes as input the security parameter* 1^λ *and outputs global system parameters* par*. We assume that* par *implicitly defines sets of public keys, secret keys, messages and signatures, respectively. All algorithms related to* MS *take* par *at least implicitly as input.*
- $\mathsf{Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$ *takes as input system parameters* par*, and outputs a public key* pk *and a secret key* sk*.*
- $\mathsf{Sig}(\mathsf{sk}, \mathsf{m}) \to \sigma$ *takes as input a secret key* sk *and a message* m *and outputs a signature* σ*.*

[6] For BLS signatures, we make the hash function H an explicit parameter, which will simplify notation later. Concretely, the proofs of possession in BLS multi-signatures are implemented using BLS on a different hash function. For our constructions, we omit adding every hash function as an explicit parameter to avoid clutter.

- $\mathsf{Comb}((\mathsf{pk}_1,\sigma_1),\ldots,(\mathsf{pk}_N,\sigma_N),\mathsf{m}) \to \sigma$ *is deterministic, takes as input a list of keys and signatures* $(\mathsf{pk}_1,\sigma_1),\ldots,(\mathsf{pk}_N,\sigma_N)$*, and a message* m*, and outputs a signature* σ*.*
- $\mathsf{Ver}(\mathsf{pk}_1,\ldots,\mathsf{pk}_N,\mathsf{m},\sigma) \to b$ *is deterministic, takes as input a list of public keys* $\mathsf{pk}_1,\ldots,\mathsf{pk}_N$*, a message* m*, and a signature* σ*, and outputs a bit* $b \in \{0,1\}$*.*

We require that MS *is complete in the following sense. For all* $\mathsf{par} \in \mathsf{Setup}(1^\lambda)$*, all* $N \in \mathbb{N}$*, all* $(\mathsf{pk}_i,\mathsf{sk}_i) \in \mathsf{Gen}(\mathsf{par})$ *for every* $i \in [N]$*, and all messages* m*, we have*

$$\Pr\left[\mathsf{Ver}(\mathsf{pk}_1,\ldots,\mathsf{pk}_N,\mathsf{m},\sigma)=1 \;\middle|\; \begin{array}{l}\forall i \in [N]:\ \sigma_i \leftarrow \mathsf{Sig}(\mathsf{sk}_i,\mathsf{m}),\\ \sigma := \mathsf{Comb}((\mathsf{pk}_1,\sigma_1),\ldots,(\mathsf{pk}_N,\sigma_N),\mathsf{m})\end{array}\right] = 1.$$

Below, we define unforgeability for multi-signatures following our syntax. As in the unforgeability game for signatures, the adversary gets access to a target public key and to a signing oracle. The only difference to the game for signatures is that the forgery is now a combined signature, and is expected to come with a list of public keys that includes the target public key.

Definition 6 (EUF-CMASecurity for Multi-Signatures). *Let* $\mathsf{MS} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Sig}, \mathsf{Comb}, \mathsf{Ver})$ *be a multi-signature scheme. Consider an adversary* $\mathcal{A}$ *and the game* $\mathbf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MS}}(\lambda)$ *defined as follows:*

1. *Run* $\mathsf{par} \leftarrow \mathsf{Setup}(1^\lambda)$ *and* $(\mathsf{pk},\mathsf{sk}) \leftarrow \mathsf{Gen}(\mathsf{par})$*.*
2. *Initialize* $\mathcal{Q} := \emptyset$ *and let* O *be the following oracle:*
   - $\mathrm{O}(\mathsf{m})$: *Take as input* m*, set* $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{m}\}$*, and return* $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk},\mathsf{m})$*.*
3. *Run* $\mathcal{A}$ *on input* $(\mathsf{par},\mathsf{pk})$ *and with access to oracle* O*. Obtain a list of keys* $(\mathsf{pk}^*_1,\ldots,\mathsf{pk}^*_N)$ *and a pair* (m^*,σ^*) *from* $\mathcal{A}$*.*
4. *Output* 1 *if and only if there is an index* $i \in [N]$ *such that* $\mathsf{pk} = \mathsf{pk}^*_i$*, and it holds that* $\mathsf{Ver}(\mathsf{pk}^*_1,\ldots,\mathsf{pk}^*_N,\mathsf{m}^*,\sigma^*) = 1$*, and that* $\mathsf{m}^* \notin \mathcal{Q}$*.*

We say that MS *is* EUF-CMA *secure, if for all PPT adversaries* $\mathcal{A}$*, the following advantage is negligible:*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{MS}}(\lambda) := \Pr\left[\mathbf{EUF\text{-}CMA}^{\mathcal{A}}_{\mathsf{MS}}(\lambda) \Rightarrow 1\right].$$

4 Variants of BLS Multi-signatures

In this section, we present our new tightly secure variant of BLS multi-signatures. To avoid repetition and to highlight the similarity to BLS, we do not only define a single scheme, but rather a class of schemes BLSMS_L, where $L \in \mathbb{N}$ is a parameter. Informally, this parameter specifies how many instances of BLS are combined. The interesting cases for this paper are as follows:

- $L = 1$: This corresponds to standard BLS multi-signatures with proofs of possession as used for example in Ethereum [26]. Here, we give the to this date tightest known proof without the algebraic group model. Essentially, the security is not larger than for (single-signer) BLS signatures.

– $L = 2$: In this case, we obtain a tightly secure multi-signature based on CDH.

Interestingly, these constructions are fully compatible, i.e., one signer may use $L = 1$ whereas a different signer may choose to use $L = 2$ or even[7] $L = 3$. As corollaries of the case $L = 2$, we obtain new tightly secure variants of (single-signer) BLS signatures.

4.1 Parameterized Construction

Let $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}_1$ be a random oracle which informally takes the role of the random oracle in BLS signatures. We first define helper algorithms $\mathsf{KeyProve}$ and KeyVer. Roughly, these are used to prove possession of secret keys, which is a common method to prevent rogue-key attacks [26]. The way they are commonly implemented is as BLS signatures on the public key itself, using a different random oracle $\mathsf{G}\colon \{0,1\}^* \to \mathbb{G}_1$:

- $\mathsf{KeyProve}(\mathsf{pk}, \mathsf{sk}) \to \pi$: Set $\pi := \mathsf{BLS.Sig}^{\mathsf{G}}(\mathsf{sk}, \mathsf{pk})$
- $\mathsf{KeyVer}(\mathsf{pk}, \pi) \to \{0,1\}$: Return $b := \mathsf{BLS.Ver}^{\mathsf{G}}(\mathsf{pk}, \mathsf{pk}, \pi)$.

Finally, we use a third random oracle $\hat{\mathsf{H}}\colon \{0,1\}^* \to \{0,\ldots,L-1\}$ that will be used to randomly decide which key to use for signing. With these algorithms at hand, we now define BLSMS_L for $L \in \mathbb{N}$.

```
– $\mathsf{BLSMS}_L.\mathsf{Setup}(1^\lambda) \to \mathsf{par}$:
    1. $\mathsf{par} \leftarrow \mathsf{BLS.Setup}(1^\lambda)$
– $\mathsf{BLSMS}_L.\mathsf{Gen}(\mathsf{par}) \to (\mathsf{pk}, \mathsf{sk})$:
    1. $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$
    2. For all $\beta \in \{0,\ldots,L-1\}$ : $(\mathsf{pk}_\beta, \mathsf{sk}_\beta) \leftarrow \mathsf{BLS.Gen}(\mathsf{par})$
    3. $\mathsf{sk} := (\mathsf{sk}_0, \ldots, \mathsf{sk}_{L-1}, \mathsf{seed})$
    4. For all $\beta \in \{0,\ldots,L-1\}$: $\pi_\beta := \mathsf{KeyProve}(\mathsf{pk}_\beta, \mathsf{sk}_\beta)$
    5. $\pi := (\pi_0, \ldots, \pi_{L-1})$,   $\mathsf{pk} := \big((\mathsf{pk}_0, \pi_0), \ldots, (\mathsf{pk}_{L-1}, \pi_{L-1})\big)$
– $\mathsf{BLSMS}_L.\mathsf{Sig}(\mathsf{sk}, \mathsf{m}) \to \sigma$:
    1. $\beta_{\mathsf{m}} := \hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m})$
    2. $\sigma := \mathsf{BLS.Sig}^{\mathsf{H}}(\mathsf{sk}_{\beta_{\mathsf{m}}}, \mathsf{m})$
– $\mathsf{BLSMS}_L.\mathsf{Comb}((\mathsf{pk}_1, \sigma_1), \ldots, (\mathsf{pk}_N, \sigma_N), \mathsf{m}) \to \sigma$:
    1. For all $i \in [N]$: parse $\mathsf{pk}_i = \big((\mathsf{pk}_{i,0}, \ldots, \mathsf{pk}_{i,L-1}), (\pi_{i,0}, \ldots, \pi_{i,L-1})\big)$
    2. For all $i \in [N]$: $\beta_i := \min\{\beta \in \{0,\ldots,L-1\} \mid \mathsf{BLS.Ver}^{\mathsf{H}}(\mathsf{pk}_{i,\beta}, \mathsf{m}, \sigma_i) = 1\}$
    3. $\bar{\sigma} := \prod_{i=1}^N \sigma_i$,   $\sigma := (\bar{\sigma}, \beta_1, \ldots, \beta_N)$
– $\mathsf{BLSMS}_L.\mathsf{Ver}(\mathsf{pk}_1, \ldots, \mathsf{pk}_N, \mathsf{m}, \sigma) \to b$:
    1. For all $i \in [N]$: parse $\mathsf{pk}_i = \big((\mathsf{pk}_{i,0}, \ldots, \mathsf{pk}_{i,L-1}), (\pi_{i,0}, \ldots, \pi_{i,L-1})\big)$
    2. Parse $\sigma = (\bar{\sigma}, \beta_1, \ldots, \beta_N)$
    3. $\bar{\mathsf{pk}} := \prod_{i=1}^N \mathsf{pk}_{i,\beta_i}$
    4. $b := \mathsf{BLS.Ver}^{\mathsf{H}}(\bar{\mathsf{pk}}, \mathsf{m}, \bar{\sigma}) \wedge \bigwedge_{i \in [N]} \mathsf{KeyVer}(\mathsf{pk}_{i,\beta_i}, \pi_{i,\beta_i})$
```

[7] One can also use our technique to prove tight security from CDH for $L = 3$, but we do not see a good reason to use this scheme, so we omit presenting this proof.

Remark 2. In many applications, one would verify the proofs $\pi_{i,\beta}$ contained in public keys once when a party registers.

Remark 3. For the case $L = 1$, this scheme is exactly the standard BLS multi-signature scheme with proofs of possesion, noting that the step $\beta_{\mathsf{m}} := \hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m})$ can safely omitted as β_{m} is fixed in this case in the signing algorithm. Similarly, the bits β_i can be omitted from the combined signature.

Remark 4. A combined signature has size $\mathsf{size}(\mathbb{G}_1) + N \log L$, where $\mathsf{size}(\mathbb{G}_1)$ denotes the size of a single group element. That is, the size of the signature still scales linearly with the number of signers N. However, for the interesting parameters $L \in \{1, 2\}$, this means at most one bit per signer. In practice, this can be ignored, as this only exceeds a small number of group elements for a very large number of signers. Constructions with similar efficiency have been proposed before [47,48].

Remark 5. One interesting feature of these multi-signatures is that they are interoperable: one signer may decide to keep using standard BLS signatures BLSMS_1, and another signer may choose to use BLSMS_2 or BLSMS_L for arbitrary $L \in \mathbb{N}$. The signatures can still be combined using obvious adaptations of algorithm Comb. It is also clear that from the perspective of a single signer using BLSMS_L to sign, the security of BLSMS_L applies even if the adversary may use a different L. For example, a signer using $L = 2$ has tight security from CDH even when other users use standard BLS multi-signatures.

4.2 Security with One Key: BLS Multi-signatures

Regular BLS multi-signatures correspond to the case $L = 1$. Ristenpart and Yilek [51] gave a proof for this variant with a security loss similar to standard BLS signatures (cf. Lemma 1). Their proof is in the type-2 pairing setting (following the well-known classification in [29]), i.e., it relies on the existence of an efficiently computable isomorphism $\psi \colon \mathbb{G}_2 \to \mathbb{G}_1$. As outlined in the technical overview, we give a proof with the same security loss without this assumption. In this way, our proof also applies to the type-3 pairing setting. We give the proof in our full version [3] and note that the proof is a simplified version of the proof of Theorem 2.

Theorem 1. *Assume that* $\mathsf{H} \colon \{0,1\}^* \to \mathbb{G}_1$*, and* $\mathsf{G} \colon \{0,1\}^* \to \mathbb{G}_1$ *are random oracles. If the* CDH *assumption holds relative to* PGGen*, then* BLSMS_1 *is* $\mathsf{EUF\text{-}CMA}$ *secure. More precisely, for every PPT algorithm* $\mathcal{A}$*, there is a PPT algorithm* $\mathcal{B}$ *with* $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ *and*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{BLSMS}_1}(\lambda) \leq \frac{8Q_S + 4Q_SQ_{\mathsf{H}} + 4Q_SQ_{\mathsf{G}}}{p} + 4Q_S \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda),$$

where $Q_{\mathsf{H}}, Q_{\mathsf{G}}, Q_S$ *denote the number of queries to* H, G*, and* O*, respectively.*

4.3 Two Keys and Tight Security

With $L = 2$, we get tight security from CDH, which is stated in the following theorem.

Theorem 2. *Assume that* $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}_1$, $\mathsf{G}\colon \{0,1\}^* \to \mathbb{G}_1$, *and* $\hat{\mathsf{H}}\colon \{0,1\}^* \to \{0,1\}$ *are random oracles. If the* CDH *assumption holds relative to* PGGen, *then* BLSMS_2 *is* $\mathsf{EUF\text{-}CMA}$ *secure. More precisely, for every PPT algorithm $\mathcal{A}$, there is a PPT algorithm $\mathcal{B}$ with $\mathbf{T}(\mathcal{A}) \approx \mathbf{T}(\mathcal{B})$ and*

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{BLSMS}_2}(\lambda) \leq \frac{Q_{\hat{\mathsf{H}}}}{2^\lambda} + \frac{8 + 4Q_{\mathsf{H}} + 4Q_{\mathsf{G}}}{p} + 4 \cdot \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda),$$

where $Q_{\mathsf{H}}, Q_{\mathsf{G}}, Q_{\hat{\mathsf{H}}}$ *denote the number of queries to* H, G, *and* $\hat{\mathsf{H}}$, *respectively.*

Proof. We present our proof as a sequence of games, where the first game $\mathbf{G}_0$ is the $\mathsf{EUF\text{-}CMA}$ game. In games $\mathbf{G}_1$ to $\mathbf{G}_2$, we guess whether the adversary forges with respect to public key pk_0 or pk_1. Say our guess is $\hat{\beta} \in \{0,1\}$, meaning we now assume that the forgery is with respect to $\mathsf{pk}_{\hat{\beta}}$. Similarly, we ensure that $\mathsf{pk}_{\hat{\beta}}$ is not the key that the honest signer would have used for the forgery message. In games $\mathbf{G}_3$ and $\mathbf{G}_4$, we embed a random group element v into random oracle outputs for oracle H and establish that we can simulate the signing oracle efficiently without using $\mathsf{sk}_{\hat{\beta}}$. Intuitively, v and $\mathsf{pk}_{\hat{\beta}}$ will correspond to the CDH instance. At this point, the proof for the single signer setting would end with a reduction to the CDH assumption. As we are in the multi-signature setting, the following steps are needed: in game $\mathbf{G}_5$, we establish that the proof of possession for $\mathsf{pk}_{\hat{\beta}}$ can be simulated without using $\mathsf{sk}_{\hat{\beta}}$. In game $\mathbf{G}_6$, we then embed v into the adversary's proofs of possession. This is essential for eliminating terms related to adversarially chosen keys in the forgery when we then reduce to CDH. Let us now make all of this more precise.

Game $\mathbf{G}_0$: This is the $\mathsf{EUF\text{-}CMA}$ game for scheme BLSMS_2 and adversary $\mathcal{A}$, with a conceptual modification in the winning condition. We recall this game to fix notation. The game does the following to generate parameters and keys:

1. Set $\mathsf{par} := (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda)$.
2. Set $\mathcal{Q} := \emptyset$ and initialize empty maps $h[\cdot], \hat{h}[\cdot]$, and $g[\cdot]$.
3. Sample $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{sk}_0, \mathsf{sk}_1 \xleftarrow{\$} \mathbb{Z}_p$.
4. Set $\mathsf{pk}_0 := g_2^{\mathsf{sk}_0}$, $\mathsf{pk}_1 := g_2^{\mathsf{sk}_1}$ and $\tilde{\mathsf{pk}}_0 := g_1^{\mathsf{sk}_0}$, $\tilde{\mathsf{pk}}_1 := g_1^{\mathsf{sk}_1}$.
5. Set $\pi_0 := \mathsf{G}(\mathsf{pk}_0)^{\mathsf{sk}_0}$ and $\pi_1 := \mathsf{G}(\mathsf{pk}_1)^{\mathsf{sk}_1}$.
6. Set $\mathsf{pk} := ((\mathsf{pk}_0, \pi_0), (\mathsf{pk}_1, \pi_1))$

Note that $\tilde{\mathsf{pk}}_0$ and $\tilde{\mathsf{pk}}_1$ are not used yet, but will be used in the following games. The game gives par and pk to the adversary $\mathcal{A}$. In addition, $\mathcal{A}$ gets access to random oracles H, $\hat{\mathsf{H}}$, G, and a signing oracle O. For the proof it will be useful that $\hat{\mathsf{H}}(\mathsf{seed}, \cdot)$ is implemented indirectly via a random oracle $\mathsf{B}\colon \{0,1\}^* \to \{0,1\}$ that is implemented lazily and only known to the game. In this way, the game will be able to distinguish queries made by $\mathcal{A}$ from queries it made itself. With this in mind, the oracles are implemented as follows:

- $\mathsf{H}(\mathsf{m})$: If $h[\mathsf{m}] = \bot$, sample $h[\mathsf{m}] \xleftarrow{\$} \mathbb{G}_1$. Return $h[\mathsf{m}]$.
- $\hat{\mathsf{H}}(\mathsf{seed}', \mathsf{m})$: If $\hat{h}[\mathsf{seed}', \mathsf{m}] = \bot$, do:
  1. If $\mathsf{seed}' = \mathsf{seed}$, set $\hat{h}[\mathsf{seed}', \mathsf{m}] := \mathsf{B}(\mathsf{m})$.
  2. Else, sample $\hat{h}[\mathsf{seed}', \mathsf{m}] \xleftarrow{\$} \{0, 1\}$.

  Return $\hat{h}[\mathsf{seed}', \mathsf{m}]$.
- $\mathsf{G}(\mathsf{pk})$: If $g[\mathsf{pk}] = \bot$, sample $g[\mathsf{pk}] \xleftarrow{\$} \mathbb{G}_1$. Return $g[\mathsf{pk}]$.
- $\mathsf{O}(\mathsf{m})$: Set $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{m}\}$, set $\beta_\mathsf{m} := \hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m})$, return $\sigma := \mathsf{H}(\mathsf{m})^{\mathsf{sk}_{\beta_\mathsf{m}}}$.

Finally, the adversary $\mathcal{A}$ outputs a list of public keys and a forgery. In the actual $\mathsf{EUF\text{-}CMA}$ game for this scheme, each such public key is a pair of BLS keys with associated proofs of possession, and the signature would contain one bit for each such pair indicating which key is used. Without loss of generality[8], we simplify the game here by assuming that $\mathcal{A}$ directly outputs a set of BLS keys with their proofs of possession. More precisely, we assume that $\mathcal{A}$ outputs a list of N pairs $(\mathsf{pk}_i^*, \pi_i^*) \in \mathbb{G}_2 \times \mathbb{G}_1$, $i \in [N]$ and a forgery $(\mathsf{m}^*, \sigma^*) \in \{0,1\}^* \times \mathbb{G}_1$. The game does the following to determine if it outputs 0 or 1:

1. If $\mathsf{m}^* \in \mathcal{Q}$, terminate with output 0.
2. Set $\mathcal{V}_0 := \{i \in [N] \mid \mathsf{pk}_i^* = \mathsf{pk}_0\}$ and $\mathcal{V}_1 := \{i \in [N] \setminus \mathcal{V}_0 \mid \mathsf{pk}_i^* = \mathsf{pk}_1\}$.
3. If $\mathcal{V}_0 \cup \mathcal{V}_1 = \emptyset$, terminate with output 0.
4. If there is an $i \in [N]$ with $\mathsf{e}\mathsf{G}(\mathsf{pk}_i^*)\mathsf{pk}_i^* \neq \mathsf{e}\pi_i^* g_2$, terminate with output 0.
5. Set $h^* := \mathsf{H}(\mathsf{m}^*)$ and $\bar{\mathsf{pk}}^* = \prod_{i=1}^N \mathsf{pk}_i^*$.
6. If $\mathsf{e}h^*\bar{\mathsf{pk}}^* \neq \mathsf{e}\sigma^* g_2$ terminate with output 0. Otherwise, terminate with output 1.

We have

$$\mathsf{Adv}^{\mathsf{EUF\text{-}CMA}}_{\mathcal{A},\mathsf{BLSMS}_2}(\lambda) \leq \Pr[\mathbf{G}_0 \Rightarrow 1].$$

Game $\mathbf{G}_1$: In this game, we change the winning condition, such that the game now additionally outputs 0 if the adversary ever queried $\hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m}^*)$. More precisely, the new check to evaluate the winning condition is as follows:

1. If $\mathsf{m}^* \in \mathcal{Q}$, terminate with output 0.
2. If $\hat{h}[\mathsf{seed}, \mathsf{m}^*] \neq \bot$, terminate with output 0.
3. Set $\mathcal{V}_0 := \{i \in [N] \mid \mathsf{pk}_i^* = \mathsf{pk}_0\}$ and $\mathcal{V}_1 := \{i \in [N] \setminus \mathcal{V}_0 \mid \mathsf{pk}_i^* = \mathsf{pk}_1\}$.
4. If $\mathcal{V}_0 \cup \mathcal{V}_1 = \emptyset$, terminate with output 0.
5. If there is an $i \in [N]$ with $\mathsf{e}\mathsf{G}(\mathsf{pk}_i^*)\mathsf{pk}_i^* \neq \mathsf{e}\pi_i^* g_2$, terminate with output 0.
6. Set $h^* := \mathsf{H}(\mathsf{m}^*)$ and $\bar{\mathsf{pk}}^* = \prod_{i=1}^N \mathsf{pk}_i^*$.
7. If $\mathsf{e}h^*\bar{\mathsf{pk}}^* \neq \mathsf{e}\sigma^* g_2$ terminate with output 0. Otherwise, terminate with output 1.

[8] A reduction can just drop the unused keys that it got from the adversary, and remove the bits from the forgery.

Here, the highlighted line is what we added. If $\mathsf{m}^* \in \mathcal{Q}$, the change has no effect. Otherwise, if $\mathbf{G}_0$ and $\mathbf{G}_1$ differ in their output, then $\mathcal{A}$ must have queried $\hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m})$ for some m, concretely, for m^*. As $\mathcal{A}$ obtains no information about seed, and seed is uniform over $\{0,1\}^\lambda$, the probability that this happens is at most $1/2^\lambda$ for each fixed random oracle query. With a union bound over all random oracle queries, we get

$$|\Pr[\mathbf{G}_0 \Rightarrow 1] - \Pr[\mathbf{G}_1 \Rightarrow 1]| \leq \frac{Q_{\hat{\mathsf{H}}}}{2^\lambda}.$$

Game $\mathbf{G}_2$: We let the game sample a bit $\hat{\beta} \xleftarrow{\$} \{0,1\}$ at the beginning, and again change the winning condition. Now, it is as follows:

1. If $\mathsf{m}^* \in \mathcal{Q}$, terminate with output 0.
2. If $\hat{h}[\mathsf{seed}, \mathsf{m}^*] \neq \bot$, terminate with output 0.
3. Set $\mathcal{V}_0 := \{i \in [N] \mid \mathsf{pk}_i^* = \mathsf{pk}_0\}$ and $\mathcal{V}_1 := \{i \in [N] \setminus \mathcal{V}_0 \mid \mathsf{pk}_i^* = \mathsf{pk}_1\}$.
4. If $\mathcal{V}_0 \cup \mathcal{V}_1 = \emptyset$, terminate with output 0.
5. Set $\beta_{\mathsf{m}^*} := \hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m}^*)$. If $\mathcal{V}_{\hat{\beta}} = \emptyset$ or $\hat{\beta} \neq 1 - \beta_{\mathsf{m}^*}$, terminate with output 0.
6. If there is an $i \in [N]$ with $\mathsf{eG}(\mathsf{pk}_i^*)\mathsf{pk}_i^* \neq \mathsf{e}\pi_i^* g_2$, terminate with output 0.
7. Set $h^* := \mathsf{H}(\mathsf{m}^*)$ and $\bar{\mathsf{pk}}^* = \prod_{i=1}^N \mathsf{pk}_i^*$.
8. If $\mathsf{e}h^*\bar{\mathsf{pk}}^* \neq \mathsf{e}\sigma^* g_2$ terminate with output 0. Otherwise, terminate with output 1.

If $\mathcal{V}_0 \cup \mathcal{V}_1 = \emptyset$ or $\hat{h}[\mathsf{seed}, \mathsf{m}^*] \neq \bot$, the change has no effect. Otherwise, note that $\mathcal{A}$'s view is independent of $\hat{\beta}$ and β_{m^*}. Therefore, we get

$$\Pr[\mathbf{G}_2 \Rightarrow 1] \geq \Pr\left[1 - \beta_{\mathsf{m}^*} = \hat{\beta} \wedge \mathcal{V}_{\hat{\beta}} \neq \emptyset\right] \cdot \Pr[\mathbf{G}_1 \Rightarrow 1].$$

As $\mathcal{V}_0 \cup \mathcal{V}_1 \neq \emptyset$, the probability of the event $\mathcal{V}_{\hat{\beta}} \neq \emptyset$ is at least $1/2$. Also, even conditioned on a fixed $\hat{\beta}$, the probability of $1 - \beta_{\mathsf{m}^*} = \hat{\beta}$ is $1/2$, as $\hat{\beta}$ and β_{m^*} are independent random variables. So we get

$$\Pr[\mathbf{G}_2 \Rightarrow 1] \geq \frac{1}{4} \cdot \Pr[\mathbf{G}_1 \Rightarrow 1].$$

Game $\mathbf{G}_3$: We change how oracle H is implemented. For this, we let the game sample an element $v \xleftarrow{\$} \mathbb{G}_1$ at the beginning. Further, the game now internally holds a random oracle $\Gamma\colon \{0,1\}^* \to \mathbb{Z}_p$, implemented lazily in the standard way, and implements H as follows:

```
– H(m) : If $h[\mathsf{m}] = \bot$, do:
    1. If $1 - \mathsf{B}(\mathsf{m}) = \hat{\beta}$, set $h[\mathsf{m}] := v^{\Gamma(\mathsf{m})}$.
    2. Else, set $h[\mathsf{m}] := {g_1}^{\Gamma(\mathsf{m})}$.
  Return $h[\mathsf{m}]$.
```

Assuming $v \neq 1 \in \mathbb{G}_1$, the outputs of H are still uniform and independent of the rest of the game and each other. The probability that $v = 1$ is at most $1/p$, so that we get[9]

$$|\Pr[\mathbf{G}_2 \Rightarrow 1] - \Pr[\mathbf{G}_3 \Rightarrow 1]| \leq \frac{1}{p}.$$

Game $\mathbf{G}_4$: We change how the signing oracle O is implemented. After this change, the secret key $\mathsf{sk}_{\hat{\beta}}$ will no longer be used by the signing oracle:

– $\mathsf{O}(\mathsf{m})$: Set $\mathcal{Q} := \mathcal{Q} \cup \{\mathsf{m}\}$, set $\beta_{\mathsf{m}} := \hat{\mathsf{H}}(\mathsf{seed}, \mathsf{m})$, and do:
  1. If $\beta_{\mathsf{m}} = \hat{\beta}$, return $\sigma := \tilde{\mathsf{pk}}_{\hat{\beta}}^{\Gamma(\mathsf{m})}$.
  2. Else, return $\sigma := \mathsf{H}(\mathsf{m})^{\mathsf{sk}_{1-\hat{\beta}}}$.

Clearly, we did not change the signing oracle for the case that $\beta_{\mathsf{m}} = 1 - \hat{\beta}$. For the other case, i.e., $\beta_{\mathsf{m}} = \hat{\beta}$, we have

$$\mathsf{H}(\mathsf{m})^{\mathsf{sk}_{\beta_\mathsf{m}}} = \left(g_1{}^{\Gamma(\mathsf{m})}\right)^{\mathsf{sk}_{\beta_\mathsf{m}}} = g_1^{\Gamma(\mathsf{m})\cdot\mathsf{sk}_{\hat{\beta}}} = \tilde{\mathsf{pk}}_{\hat{\beta}}^{\Gamma(\mathsf{m})}.$$

Therefore, we get

$$\Pr[\mathbf{G}_3 \Rightarrow 1] = \Pr[\mathbf{G}_4 \Rightarrow 1].$$

Game $\mathbf{G}_5$: We change how the game computes $\pi_{\hat{\beta}}$. Namely, from now on, the initial steps of the game are:

1. Sample $\hat{\beta} \xleftarrow{\$} \{0,1\}$ and $v \xleftarrow{\$} \mathbb{G}_1$ and set $\mathsf{par} := (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda)$.
2. Set $\mathcal{Q} := \emptyset$ and initialize empty maps $h[\cdot], \hat{h}[\cdot]$, and $g[\cdot]$.
3. Sample $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda$ and $\mathsf{sk}_0, \mathsf{sk}_1 \xleftarrow{\$} \mathbb{Z}_p$.
4. Set $\mathsf{pk}_0 := g_2^{\mathsf{sk}_2}$, $\mathsf{pk}_1 := g_2^{\mathsf{sk}_1}$ and $\tilde{\mathsf{pk}}_0 := g_1^{\mathsf{sk}_0}$, $\tilde{\mathsf{pk}}_1 := g_1^{\mathsf{sk}_1}$.
5. Sample $\delta \xleftarrow{\$} \mathbb{Z}_p$ and set $g[\mathsf{pk}_{\hat{\beta}}] := g_1^\delta$.
6. Set $\pi_{1-\hat{\beta}} := \mathsf{G}(\mathsf{pk}_{1-\hat{\beta}})^{\mathsf{sk}_{1-\hat{\beta}}}$ and $\pi_{\hat{\beta}} := \tilde{\mathsf{pk}}_{\hat{\beta}}^\delta$.
7. Set $\mathsf{pk} := ((\mathsf{pk}_0, \pi_0), (\mathsf{pk}_1, \pi_1))$

It is easy to see that the distribution of the outputs of G remains unchanged, as δ is sampled uniformly. Also, the distribution of $\pi_{\hat{\beta}}$ is not changed because

$$\mathsf{G}(\mathsf{pk}_{\hat{\beta}})^{\mathsf{sk}_{\hat{\beta}}} = g_1^{\delta\mathsf{sk}_{\hat{\beta}}} = \tilde{\mathsf{pk}}_{\hat{\beta}}^\delta.$$

We get

$$\Pr[\mathbf{G}_4 \Rightarrow 1] = \Pr[\mathbf{G}_5 \Rightarrow 1].$$

Note that the game can now be simulated without ever using $\mathsf{sk}_{\hat{\beta}}$, assuming $\mathsf{pk}_{\hat{\beta}}$ and $\tilde{\mathsf{pk}}_{\hat{\beta}}$ are given.

[9] For the interested reader, this is where we use the indirection given by implementing $\hat{\mathsf{H}}(\mathsf{seed}, \cdot)$ via B. If we would have queried $\hat{\mathsf{H}}$ in the implementation of H, then $\mathcal{A}$ could easily make the output of $\mathbf{G}_2$ and $\mathbf{G}_3$ differ by querying $\mathsf{H}(\mathsf{m}^*)$.

Game $\mathbf{G}_6$: We change oracle G. Namely, the game now internally holds a random oracle $\varDelta\colon \{0,1\}^* \to \mathbb{Z}_p$, implemented lazily in the standard way, and implements G as follows:

– $\mathsf{G}(\mathsf{pk})$: If $g[\mathsf{pk}] = \bot$, set $g[\mathsf{pk}] := v^{\varDelta(\mathsf{pk})}$. Return $g[\mathsf{pk}]$.

If $v \neq 1 \in \mathbb{G}_1$, the distribution remains unchanged, and the probability that $v = 1$ is at most $1/p$, so

$$|\Pr[\mathbf{G}_5 \Rightarrow 1] - \Pr[\mathbf{G}_6 \Rightarrow 1]| \leq \frac{1}{p}.$$

Final Reduction: Before we give the final reduction breaking CDH, we examine the forgery signature σ^* after the changes we have made and provide intuition how the reduction can solve CDH. To this end, assume $\mathbf{G}_6$ outputs 1. For ease of notation, set $x := \mathsf{sk}_{\hat{\beta}}$ and let $y \in \mathbb{Z}_p$ be such that $v = g_1^y$. We assume that a reduction is given g_1^x, g_2^x, and g_1^y, and its goal is to output g_1^{xy}. By the changes we have made, such a reduction never explicitly needs x or y over $\mathbb{Z}_p$ to simulate $\mathbf{G}_6$. Recall that the final forgery (m^*, σ^*) of the adversary comes with a list of pairs $(\mathsf{pk}_i^*, \pi_i^*)$ for $i \in [N]$. Denote by $\mathcal{C} \subseteq [N]$ the set of indices i such that $\mathsf{pk}_i^* \neq \mathsf{pk}_{\hat{\beta}}$. Intuitively, this corresponds to the set of indices for which the list contains adversarially chosen public keys. For each $i \in \mathcal{C}$, let $\mathsf{sk}_i^* \in \mathbb{Z}_p$ be such that $\mathsf{pk}_i^* = g_2^{\mathsf{sk}_i^*}$. We set $\ell := |[N] \setminus \mathcal{C}|$. Due to the change in $\mathbf{G}_2$, we know that $\ell \neq 0$. By the verification equation, we know that

$$\sigma^* = \mathsf{H}(\mathsf{m}^*)^{\bar{\mathsf{sk}}^*} \text{ for } \bar{\mathsf{sk}}^* = \ell x + \sum_{i \in \mathcal{C}} \mathsf{sk}_i^*.$$

By definition of $\mathsf{H}(\mathsf{m}^*)$ (see $\mathbf{G}_3$) and recalling that $1 - \beta_{\mathsf{m}^*} = \hat{\beta}$ (see $\mathbf{G}_2$), we know that the discrete logarithm of σ^* with respect to g_1 is

$$\varGamma(\mathsf{m}^*) \cdot y \cdot \bar{\mathsf{sk}}^* = \varGamma(\mathsf{m}^*) \cdot y \cdot \left(\ell x + \sum_{i \in \mathcal{C}} \mathsf{sk}_i^*\right) = \varGamma(\mathsf{m}^*) \cdot \left(\ell xy + y \cdot \sum_{i \in \mathcal{C}} \mathsf{sk}_i^*\right).$$

Rearranging and taking to the exponent yields

$$g_1^{xy} = \left(\sigma^{*\,1/\varGamma(\mathsf{m}^*)} \cdot \prod_{i \in \mathcal{C}} g_1^{-y\mathsf{sk}_i^*}\right)^{1/\ell}, \tag{1}$$

assuming $\varGamma(\mathsf{m}^*) \neq 0$ for now. Further, for every $i \in \mathcal{C}$, we get from the definition of G (see $\mathbf{G}_6$), from $v = g_1^y$, and from the verification equation that

$$\pi_i^* = \mathsf{G}(\mathsf{pk}_i^*)^{\mathsf{sk}_i^*} = g_1^{y\varDelta(\mathsf{pk}_i^*)\mathsf{sk}_i^*}. \tag{2}$$

Assuming $\varDelta(\mathsf{pk}_i^*) \neq 0$, rearranging Eq. (2) and plugging it into Eq. (1), we get

$$g_1^{xy} = \left(\sigma^{*\,1/\varGamma(\mathsf{m}^*)} \cdot \prod_{i \in \mathcal{C}} \pi_i^{*\,-1/\varDelta(\mathsf{pk}_i^*)}\right)^{1/\ell}. \tag{3}$$

Now our main observation is that the right-hand side of Eq. (3) can efficiently be computed. We now turn this into a reduction $\mathcal{B}$ solving the CDH problem if $\mathbf{G}_6$ outputs 1 and assuming $\Gamma(\mathsf{m}^*) \neq 0$ and $\Delta(\mathsf{pk}_i^*) \neq 0$:

1. The reduction $\mathcal{B}$ gets as input parameters $\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e$ and elements $X_1 = g_1^x$, $X_2 = g_2^x$, and $Y = g_1^y$. Its goal is to output g_1^{xy}.
2. The reduction sets $\mathsf{par} := (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, e) \leftarrow \mathsf{PGGen}(1^\lambda)$, samples $\hat{\beta} \overset{\$}{\leftarrow} \{0,1\}$ as in $\mathbf{G}_6$, and defines $\mathsf{pk}_{\hat{\beta}} := X_2$, $\tilde{\mathsf{pk}}_{\hat{\beta}} := X_1$, and $v := Y$. It sets up the remaining parameters and then simulates $\mathbf{G}_6$ for $\mathcal{A}$, which is possible efficiently without the knowledge of x and y.
3. When $\mathcal{A}$ outputs a forgery and $\mathbf{G}_6$ would output 1, the reduction aborts if $\Gamma(\mathsf{m}^*) = 0$ or $\Delta(\mathsf{pk}_i^*) = 0$ for any $i \in \mathcal{C}$. Otherwise, it computes g_1^{xy} as in Eq. (3) and outputs it.

We see that $\mathcal{B}$ perfectly simulates game $\mathbf{G}_6$ for $\mathcal{A}$ and the running time of $\mathcal{B}$ is dominated by the running time of $\mathcal{A}$. The probability that $\mathcal{B}$ has to abort before using Eq. (3) is at most $(Q_\mathsf{H} + Q_\mathsf{G})/p$. By the discussion above, we get

$$\Pr\left[\mathbf{G}_6 \Rightarrow 1\right] \leq \frac{Q_\mathsf{H} + Q_\mathsf{G}}{p} + \mathsf{Adv}^{\mathsf{CDH}}_{\mathcal{B},\mathsf{PGGen}}(\lambda).$$

□

Corollary 1. *Consider the digital signature scheme obtained by fixing $N = 1$ signer in the multi-signature scheme* BLSMS_2. *If the* CDH *assumption holds relative to* PGGen*, then this scheme is* EUF-CMA *secure, with a tight proof.*

5 Application: PoS Blockchains with Opt-In Tightness

We anticipate that the insights presented in this paper will prove valuable in the context of proof-of-stake (PoS) blockchain systems utilizing BLS multi-signatures, such as Ethereum [26]. Within this domain, the interoperability of our construction with regular BLS (see Remark 5) makes it particularly advantageous.

PoS Blockchains and Multi-signatures. Let us begin by revisiting, in simplified terms, the relevant aspects of a proof-of-stake blockchain. In such a system, participants can lock (aka stake) a designated quantity of coins and publicly declare a BLS public key to register as *validators*. When a block is proposed, it has to be attested by enough validators to be deemed valid. These attestations consist of BLS signatures of the block with respect to the validators' keys. What ends up on chain is the combined BLS multi-signature.

Our Multi-signatures for Opt-In Tightness. Now envision a system in which each validator can register a (small) number of BLS keys, such that signing with one of these keys means the validator attested the block. A natural motivation to use this setup is to ensure that validators can continue functioning even if access to some keys is lost. We argue that such a system allows for opt-in tightness without mandating a departure from BLS: Namely, consider Alice taking the role of a validator. As she prioritizes concrete security, she would register two BLS keys. Whenever she had to sign a block, she would pseudorandomly decide which key to use, thereby implementing our scheme. Note that she can do so by implementing minimal logic on top of existing BLS implementations. Conversely, if Bob prefers to adhere to regular BLS signatures, he can simply register a single key as usual. Importantly, Alice's signatures and Bob's remain compatible and can be aggregated seamlessly, even without awareness of Alice's adoption of our scheme. Thus, each validator retains the autonomy to *independently* select their preferred variant.

We remark that while some proof-of-stake chains indeed allow multiple BLS keys [10] and can directly implement the setting above, Ethereum currently does not[10]. Consequently, our research can be viewed as an argument for the integration of such functionality in the future.

References

1. 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, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_7
2. 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
3. Bacho, R., Wagner, B.: Tightly secure non-interactive BLS multi-signatures. Cryptology ePrint Archive, Paper 2024/1368 (2024), https://eprint.iacr.org/2024/1368
4. Bader, C., Jager, T., Li, Y., Schäge, S.: On the impossibility of tight cryptographic reductions. In: Fischlin, M., Coron, J.S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 273–304. Springer, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_10
5. Bagherzandi, A., Cheon, J.H., Jarecki, S.: Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In: Ning, P., Syverson, P.F., Jha, S. (eds.) ACM CCS 2008. pp. 449–458. ACM Press (Oct 2008). https://doi.org/10.1145/1455770.1455827
6. 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
7. Bellare, M., Dai, W.: Chain reductions for multi-signatures and the HBMS scheme. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part IV. LNCS, vol. 13093, pp. 650–678. Springer, Heidelberg (Dec 2021). https://doi.org/10.1007/978-3-030-92068-5_22
8. Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. In: Arge, L., Cachin, C., Jurdzinski, T., Tarlecki, A. (eds.) ICALP 2007. LNCS,

[10] One potential workaround would involve registering multiple validators. However, this approach necessitates locking twice the amount of funds as previously required, underscoring the potential benefits of a native support for multiple keys.

vol. 4596, pp. 411–422. Springer, Heidelberg (Jul 2007). https://doi.org/10.1007/978-3-540-73420-8_37

9. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 390–399. ACM Press (Oct / Nov 2006).https://doi.org/10.1145/1180405.1180453
10. Blockchain, H.: Harmony – Creating A Validator. https://docs.harmony.one/home/network/validators/creating-a-validator (2022), accessed: 2024-05-07
11. Blum, E., Leung, D., Loss, J., Katz, J., Rabin, T.: Analyzing the real-world security of the algorand blockchain. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 830–844. ACM Press (Nov 2023). https://doi.org/10.1145/3576915.3623167
12. 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
13. Boneh, D., Boyen, X., Goh, E.J.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (May 2005). https://doi.org/10.1007/11426639_26
14. Boneh, D., Drijvers, M., Neven, G.: Compact multi-signatures for smaller blockchains. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 435–464. Springer, Heidelberg (Dec 2018). https://doi.org/10.1007/978-3-030-03329-3_15
15. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (May 2003).https://doi.org/10.1007/3-540-39200-9_26
16. Boneh, D., Gorbunov, S., Wahby, R.S., Wee, H., Wood, C.A., Zhang, Z.: BLS Signatures. Internet-Draft draft-irtf-cfrg-bls-signature-05, Internet Engineering Task Force (Jun 2022), https://datatracker.ietf.org/doc/draft-irtf-cfrg-bls-signature/05/, work in Progress
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 (Dec 2001). https://doi.org/10.1007/3-540-45682-1_30
18. Boschini, C., Takahashi, A., Tibouchi, M.: MuSig-L: Lattice-based multi-signature with single-round online phase. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 276–305. Springer, Heidelberg (Aug 2022).https://doi.org/10.1007/978-3-031-15979-4_10
19. contributors, C.: Chia network: Implementation of bls signatures. GitHub repository (Nov 2022), https://github.com/Chia-Network/node-chia-bls, the green cryptocurrency with Chialisp
20. Coron, J.S.: On the exact security of full domain hash. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 229–235. Springer, Heidelberg (Aug 2000). https://doi.org/10.1007/3-540-44598-6_14
21. Coron, J.S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (Apr / May 2002). https://doi.org/10.1007/3-540-46035-7_18
22. 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

23. Damgård, I., Orlandi, C., Takahashi, A., Tibouchi, M.: Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 99–130. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75245-3_5
24. 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
25. Drijvers, M., Gorbunov, S., Neven, G., Wee, H.: Pixel: Multi-signatures for consensus. In: Capkun, S., Roesner, F. (eds.) USENIX Security 2020. pp. 2093–2110. USENIX Association (Aug 2020)
26. Edgington, B.: Upgrading Ethereum - A technical handbook on Ethereum's move to proof of stake and beyond. Edition 0.3: Capella [wip] edn. (2023), https://eth2book.info/capella/part3/helper/crypto/#bls-signatures
27. 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
28. Fukumitsu, M., Hasegawa, S.: A tightly secure ddh-based multisignature with public-key aggregation. In: 2020 Eighth International Symposium on Computing and Networking Workshops (CANDARW). pp. 321–327 (2020). https://doi.org/10.1109/CANDARW51189.2020.00069
29. Galbraith, S., Paterson, K., Smart, N.: Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165 (2006), https://eprint.iacr.org/2006/165
30. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: Scaling byzantine agreements for cryptocurrencies. In: Proceedings of the 26th Symposium on Operating Systems Principles. p. 51–68. SOSP '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3132747.3132757, https://doi.org/10.1145/3132747.3132757
31. Goh, E.J., Jarecki, S., Katz, J., Wang, N.: Efficient signature schemes with tight reductions to the Diffie-Hellman problems. Journal of Cryptology **20**(4), 493–514 (Oct 2007). https://doi.org/10.1007/s00145-007-0549-3
32. Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing **17**(2), 281–308 (Apr 1988)
33. Inc., C.N.: Chialisp primer: 5. bls signatures (2024), https://chialisp.com/chialisp-bls-signatures/
34. Itakura, K., Nakamura, K.: A public-key cryptosystem suitable for digital multisignatures. NEC Research & Development (71), 1–8 (1983)
35. Kakvi, S.A., Kiltz, E.: Optimal security proofs for full domain hash, revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 537–553. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_32
36. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) ACM CCS 2003. pp. 155–164. ACM Press (Oct 2003). https://doi.org/10.1145/948109.948132
37. Langford, S.K.: Weakness in some threshold cryptosystems. In: Koblitz, N. (ed.) CRYPTO'96. LNCS, vol. 1109, pp. 74–82. Springer, Heidelberg (Aug 1996). https://doi.org/10.1007/3-540-68697-5_6

38. Li, C.M., Hwang, T., Lee, N.Y.: Threshold-multisignature schemes where suspected forgery implies traceability of adversarial shareholders. In: Santis, A.D. (ed.) EUROCRYPT'94. LNCS, vol. 950, pp. 194–204. Springer, Heidelberg (May 1995). https://doi.org/10.1007/BFb0053435
39. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential aggregate signatures and multisignatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_28
40. Maxwell, G., Poelstra, A., Seurin, Y., Wuille, P.: Simple schnorr multi-signatures with applications to bitcoin. Designs, Codes and Cryptography **87**, 2139 – 2164 (2019), https://api.semanticscholar.org/CorpusID:4053539
41. Micali, S., Ohta, K., Reyzin, L.: Accountable-subgroup multisignatures: Extended abstract. In: Reiter, M.K., Samarati, P. (eds.) ACM CCS 2001. pp. 245–254. ACM Press (Nov 2001). https://doi.org/10.1145/501983.502017
42. Michels, M., Horster, P.: On the risk of disruption in several multiparty signature schemes. In: Kim, K., Matsumoto, T. (eds.) Advances in Cryptology — ASIACRYPT '96. pp. 334–345. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)
43. 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
44. Nick, J., Ruffing, T., Seurin, Y., Wuille, P.: MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1717–1731. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3417236
45. Ohta, K., Okamoto, T.: A digital multisignature scheme based on the Fiat-Shamir scheme. In: Imai, H., Rivest, R.L., Matsumoto, T. (eds.) ASIACRYPT'91. LNCS, vol. 739, pp. 139–148. Springer, Heidelberg (Nov 1993). https://doi.org/10.1007/3-540-57332-1_11
46. Organization, D.: Drand - a distributed randomness beacon daemon. GitHub repository (2020), https://github.com/drand/drand
47. Pan, J., Wagner, B.: Chopsticks: Fork-free two-round multi-signatures from non-interactive assumptions. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 597–627. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_21
48. Pan, J., Wagner, B.: Toothpicks: More efficient fork-free two-round multi-signatures. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part I. LNCS, vol. 14651, pp. 460–489. Springer, Heidelberg, Zurich, Switherland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58716-0_16
49. Qian, H., Li, X., Huang, X.: Tightly secure non-interactive multisignatures in the plain public key model. Informatica (Vilnius) **3** (01 2012). https://doi.org/10.15388/Informatica.2012.369
50. Qian, H., Xu, S.: Non-interactive multisignatures in the plain public-key model with efficient verification. Information Processing Letters **111**(2), 82–89 (2010). https://doi.org/10.1016/j.ipl.2010.10.015, https://www.sciencedirect.com/science/article/pii/S0020019010003212
51. Ristenpart, T., Yilek, S.: The power of proofs-of-possession: Securing multiparty signatures against rogue-key attacks. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 228–245. Springer, Heidelberg (May 2007). https://doi.org/10.1007/978-3-540-72540-4_13

52. Tessaro, S., Zhu, C.: Threshold and multi-signature schemes from linear hash functions. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 628–658. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_22
53. Waters, B.R.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (May 2005). https://doi.org/10.1007/11426639_7

Extractable Witness Encryption for KZG Commitments and Efficient Laconic OT

Nils Fleischhacker[1(✉)], Mathias Hall-Andersen[2], and Mark Simkin[3]

[1] Ruhr University Bochum, Bochum, Germany
mail@nilsfleischhacker.de
[2] ZkSecurity, Bochum, Germany
mathias@zksecurity.xyz
[3] Bochum, Germany
mark@univariate.org

Abstract. We present a concretely efficient and simple extractable witness encryption scheme for KZG polynomial commitments. It allows to encrypt a message towards a triple $(\mathsf{com}, \alpha, \beta)$, where com is a KZG commitment for some polynomial f. Anyone with an opening for the commitment attesting $f(\alpha) = \beta$ can decrypt, but without knowledge of a valid opening the message is computationally hidden. Our construction is simple and highly efficient. The ciphertext is only a single group element. Encryption and decryption both require a single pairing evaluation and a constant number of group operations.

Using our witness encryption scheme, we construct a simple and highly efficient laconic OT protocol, which significantly outperforms the state of the art in most important metrics.

1 Introduction

The polynomial commitment scheme of Kate, Zaverucha, and Goldberg (KZG) [24] is a powerful tool that has allowed for constructing a variety of advanced cryptographic primitives. Many concretely efficient vector commitments [8] with all kinds of additional functionalities or security properties are extensions of the KZG polynomial commitment scheme [22,25,28,29,32]. Many of the currently most efficient proof systems [9,16,26] make crucial use of KZG commitments as part of their constructions.

In general, a polynomial commitment scheme allows for committing to a polynomial $f(X)$, such that one can later provide openings, attesting that $f(\alpha) = \beta$ for some chosen evaluation point α. The polynomial commitment scheme should ensure that the committed polynomial is at most of some degree d, that it is position binding in the sense that one cannot open the commitment to two different evaluations at the same point α, and in some cases it may also be desirable that the commitment itself hides the committed polynomial.

M. Simkin—Independent Researcher.

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972.

K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15485, pp. 423–453, 2025.
https://doi.org/10.1007/978-981-96-0888-1_14

The KZG polynomial commitment scheme is highly efficient as both commitments and openings consists of a single group element, no matter how large the degree of the polynomial is. From a security perspective, the construction requires a trusted setup and can be proven secure in the algebraic group model (AGM) [15] under a q-type variant of the standard discrete logarithm assumption.

Given the widespread usefulness of KZG commitments, it seems natural to ask the following somewhat abstract questions: Can we expand the toolkit surrounding KZG commitments in a fundamental way that would allow us to solve an even larger set of problems via these polynomial commitments?

1.1 Our Contribution

In this work, we present a simple, concretely efficient extractable witness encryption [19,21] scheme for KZG commitments and show how it can be used to construct concretely efficient laconic OT [10]. Concretely, we make the following contributions:

Witness Encryption for KZG Commitments. We present an encryption scheme that takes a KZG commitment $\mathtt{com}$, evaluation point α, evaluation β, as well as a message m as input and produces ciphertext $\mathtt{ct}$. Correctness of the encryption scheme allows anybody, who knows an opening for $\mathtt{com}$ that attests that the committed polynomial evaluates to β at point α, to decrypt the ciphertext $\mathtt{ct}$. Informally, security of the encryption scheme ensures that anybody, who can decrypt $\mathtt{ct}$, can also compute a valid opening with respect to $(\mathtt{com}, \alpha, \beta)$. Now, if $\mathtt{com}$ is a commitment to polynomial $f(X)$ and $\mathtt{ct}$ is an encryption for $(\mathtt{com}, \alpha, \beta)$, where $\beta \neq f(\alpha)$, then $\mathtt{ct}$ computationally hides the message, due to the evaluation binding property of the polynomial commitment scheme.

We note that for *any* tuple $(\mathtt{com}, \alpha, \beta)$, there *exists* an opening and for this reason the vanilla notion of witness encryption as defined by Garg et al. [19] would not suffice to guarantee that the message is hidden. By focusing on extractable witness encryption as defined by Goldwasser et al. [21], we can ensure that the message is hidden, whenever computing the opening is assumed to be hard.

Our encryption scheme is highly efficient in terms of both ciphertext size and computational costs. One ciphertext for the scheme as described above is comprised of a single group element, encryption requires two group operations and one pairing evaluation and decryption requires a single pairing evaluation. The construction is generalized for multiple opening constraints. In that case the costs grow linearly in the number of constraints.

Efficient Laconic Oblivious Transfer. Using our witness encryption scheme for KZG commitments, we present a simple and highly-efficient protocol for laconic oblivious transfer [10]. Oblivious transfer [27] is an interactive protocol between a sender holding messages m_0 and m_1 and a receiver holding a choice bit b. At the end of the protocol, the receiver should only learn m_b and the sender should learn nothing at all.

In laconic oblivious transfer, the receiver holds a database $D \in \{0,1\}^n$ of n choice bits and publishes a digest $\texttt{digest} \leftarrow \mathsf{H}(D)$, whose size is independent of the size of D. The sender can then repeatedly choose a message pair (m_0, m_1), an index $i \in [n]$, and use the digest to compute a short message for the receiver, which allows them to obtain $m_{D[i]}$. As in regular OT, the receiver does not learn anything about the other message and the sender does not learn anything about the choice bit of the receiver.

Cho et al. [10] show that laconic OT is a useful building block for constructing secure computation protocols that operate on large inputs and for constructing multi-hop homomorphic encryption for RAM programs. Improving the efficiency of laconic OT, directly translates to faster protocols for these applications.

Implementation and Benchmarks. The full laconic OT construction was implemented[1] in Rust. We provide benchmarks for various parameter regimes and show that our laconic OT construction outperforms the state of the art in most important metrics quite significantly.

1.2 Related Work

In the following, let us discuss related works from several research areas that intersect with our work here.

Witness Encryption. Witness encryption was originally introduced by Garg et al. [19], who presented a construction based on multilinear maps. Such encryption schemes take a statement $\mathtt{x}$ and a message m as input and produce a ciphertext $\mathtt{ct}$. Correctness guarantees that given a witness $\mathtt{w}$ with $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}$, one can decrypt $\mathtt{ct}$ and security ensures that the message is computationally hidden, if $\mathtt{x} \notin \mathcal{L}$. It is important to note though that $\mathtt{ct}$ provides no explicit security guarantees, when $\mathtt{x} \in \mathcal{L}$. In a different work, Garg et al. [17] showed that one can construct general-purpose witness encryption from indistinguishability obfuscation [2]. Two recent works by Tsabary [30] and Vaikuntanathan [31] present construction based on new computational hardness assumptions. Extractable witness encryption is a strengthening of the original notion that was introduced by Goldwasser et al. [21]. It requires the existence of an extractor, which can recover a witness $\mathtt{w}$ from any adversary that can break the semantic security of the encryption scheme. How to construct general-purpose extractable witness encryption is currently unclear. Garg et al. [18] show that the existence of a type of special-purpose obfuscation would imply the non-existence of general-purpose extractable witness encryption. Benhamouda and Lin [4] introduce the notion of witness selectors, which lie somewhere between extractable and non-extractable

[1] https://github.com/rot256/research-we-kzg.

witness encryption. They require semantic security of the encryption to hold if finding a witness for the relation is computationally hard.

While remotely practical general-purpose witness encryption from standard assumptions currently seems out of reach, there are several works [3–5, 7, 20] that construct different types of witness encryption for specific languages. In the work of Garg and Srinivasan [20], they construct a primitive they call homomorphic proof commitments with encryption. Given a commitment $\mathtt{com}$ and a message m they allow for encrypting the message, such that a proof π that $\mathtt{com}$ is a commitment to 0 or 1 can be used to decrypt the message. Benhamouda and Lin [5] introduce witness encryption for NIZKs of commitments. Their scheme allows for encrypting towards a tuple $(\mathtt{com}, G, y)$, where $\mathtt{com}$ is a commitment, G is a function, and y is an output. The witness for decryption is a non-interactive zero-knowledge proof for the statement "the message committed in $\mathtt{com}$ is m and $G(m) = y$". In the presented construction, the commitment and proof size are at least linear in the statement size. Campanelli, Fiore, and Khoshakhlagh [7] present succinct functional commitments with an associated witness encryption scheme with short commitments and short openings. They introduce a new computational hardness assumption and use it to prove their construction secure in the AGM.

In contrast to these works, we construct an extractable witness encryption scheme for a polynomial, rather than functional, commitment scheme from standard assumptions in the AGM. We rely on KZG commitments which have succinct commitments and openings and the ciphertexts of our construction are equally succinct. Our construction is surprisingly simple and in particular much simpler than the constructions discussed above. We believe this to be a significant advantage, when it comes to engineering and deploying a cryptographic primitive in the real world.

Laconic OT. The concept of laconic OT was originally introduced in a work by Cho et al. [10]. The authors presented a construction based on a special type of encryption scheme in combination with garbled circuits. While asymptotically interesting, their construction makes non-blackbox use of cryptographic objects and is completely impractical when it comes to concrete performance.

Green, Jain, Van Laer [23] focus on concretely efficient laconic OT protocols, but the security of their proposed construction relies on a new hardness assumption they introduce. In addition, they require a generic and somewhat expensive transformation via garbled circuits to achieve sender privacy. Döttling et al. [13] present another laconic OT protocol based on the learning with errors (LWE) with error leakage problem. They show that this problem reduces to standard LWE for certain parameter ranges.

From a concrete efficiency perspective, our construction is highly efficient in terms of bandwidth and computational overheads that are induced by the individual OT executions. In comparison the work of Green, Jain, Van Laer, our public parameters are smaller by a factor of 3-4x, our sender's message size is smaller by a factor $\approx$ 5x, and our receiver's computation time is smaller by 1-6 *orders of magnitude*, depending on the precise setting. In comparison to the work of Döttling et al., our sender's OT message is smaller by 5 *orders of magnitude*. We note that reducing the sender's message size is of crucial importance, as this is the message that needs to be sent once for every single OT invocation. Our results are particularly surprising considering how much simpler our construction is compared to prior works. We provide a detailed discussion of the concrete efficiency of our construction in Sect. 6.

The main *disadvantage* of our construction is that it requires the same "powers of tau" trusted setup as the KZG commitment scheme. This setup needs to be generated either by a single trusted party or by multiple separate parties via secure computation. While needing a, rather large, trusted setup is certainly a drawback, we note that precisely this type of distributed setup generation has already been successfully performed in practice on a large scale with thousands of participants[2] and these setups *can* be reused for our scheme. We leave achieving the same efficiency, without relying on a large trusted setup as an exciting open direction for future work.

Finally, we would like to note that the laconic private set intersection construction of Aranha et al. [1] shares similarities with our final laconic OT construction in this work. It is conceivable that their construction could be modified into a labelled laconic private set intersection, which would then allow for transforming it into a laconic OT construction. Even assuming all of this being true, the resulting construction would require a setup that is at least twice as large as ours and it would only provide security against a semi-honest sender, whereas our construction is secure even when the sender is actively corrupt.

2 Preliminaries

In this section, we will recall some notation and standard definitions that we will use throughout the paper.

Notation. We denote by $\lambda \in \mathbb{N}$ the security parameter, by $\mathsf{poly}(\lambda)$ any function that is bounded by a polynomial in λ and by $\mathsf{negl}(\lambda)$ any negligible function in λ. An algorithm is PPT if it is modeled by a probabilistic Turing machine with a running time bounded by $\mathsf{poly}(\lambda)$. Let S be a set. We write $x \leftarrow S$ for the process of sampling an element of S uniformly at random. For $n \in \mathbb{N}$, we write $[n]$ to denote the set $\{1, \ldots, n\}$. For a vector $\boldsymbol{v} \in S^n$ and $i \in [n]$, we write v_i to denote its i-th component.

2.1 Algebraic Group Model

Some of our proofs will rely on the algebraic group model, which was introduced by Fuchsbauer, Kiltz, and Loss [15]. In this model, one considers algebraic algorithms and, in particular, an algebraic adversary. Let $\mathbb{G}$ of prime order

[2] https://ceremony.ethereum.org.

q. Whenever an algorithm returns a value $Y \in \mathbb{G}$, it must also provide an algebraic representation of that element $(e_1, \ldots, e_n)$, such that $Y = \prod_{n}^{i=1} X_i^{e_i}$, where $(X_1, \ldots, X_n)$ are group elements the algorithm has previously seen.

2.2 Pairings and Assumptions

Let GGen be the parameter generation algorithm that takes the security parameter λ as input and returns $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, q, e)$ as its output, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are groups of prime order $p = p(\lambda)$, where $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are generators and where $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map. For $i \in \{1, 2, T\}$, we write $[\alpha]_i$ as a shorthand notation for g_i^α and we write $[\alpha]_i + [\beta]_i = [\alpha + \beta]_i$ as a shorthand notation for the multiplication of two group elements. The type of pairing is irrelevant for our applications.

Definition 1 (ℓ-DLOG Assumption). *The ℓ-DLOG assumption holds with respect to* GGen*, if for any $\lambda \in \mathbb{N}$ and any PPT adversary $\mathcal{A}$, it holds that*

$$\Pr\left[\tau = \tau' : \begin{array}{r} \mathsf{par} \leftarrow \mathsf{GGen}(1^\lambda), \tau \leftarrow \mathbb{F}_p \\ \tau' \leftarrow \mathcal{A}(\mathsf{par}, ([\tau^0]_1, \ldots, [\tau^\ell]_1, [1]_2, [\tau]_2)) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where the probability is taken over the random coins of the group generation algorithm and the adversary and over the uniform choice of τ.

Remark 1. Throughout the paper we will omit explicitly spelling out the group parameter generation and we will assume all involved parties are always implicitly provided with the parameters.

2.3 Polynomial Commitments

A polynomial commitment allows for computing a short value $\mathtt{com}$ for a polynomial f of potentially high degree over a finite field $\mathbb{F}$. Later on, one can compute short openings that certify that the polynomial committed to by $\mathtt{com}$ evaluates to $\beta \in \mathbb{F}$ at some position $\alpha \in \mathbb{F}$. Polynomial commitment should be binding in the sense that it should be impossible to open the same point to two different values.

Definition 2 (Polynomial Commitments). *A polynomial commitment over $\mathbb{F}$ is a tuple of PPT algorithms* $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *defined as follows:*

$\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$: *The setup algorithm takes security parameter λ and degree upper bound d as input and returns commitment key* ck.

$\mathtt{com} \leftarrow \mathsf{Commit}(\mathsf{ck}, f)$: *The commitment algorithm takes commitment key* ck *and polynomial $f \in \mathbb{F}[X]$ as input and returns commitment* $\mathtt{com}$.

$\pi \leftarrow \mathsf{Open}(\mathsf{ck}, f, \alpha, \beta)$: *The opening algorithm takes commitment key* ck*, polynomial $f \in \mathbb{F}[X]$, and points $\alpha, \beta \in \mathbb{F}$ as input and returns openings π.*

$b \leftarrow \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi, \alpha, \beta)$**:** *The verification algorithm takes commitment key* ck*, commitment* $\mathtt{com}$*, opening* π*, and points* $\alpha, \beta \in \mathbb{F}$ *as input and returns a bit* b*.*

Definition 3 (Correctness). *A polynomial commitment scheme* $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *over* $\mathbb{F}$ *is correct, if for all* $\lambda, d \in \mathbb{N}$*, all* $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^d)$*, all polynomials* $f \in \mathbb{F}[X]$ *of degree at most* d*, all* $\mathtt{com} \leftarrow \mathsf{Commit}(\mathsf{ck}, f)$*, all* $\alpha, \beta \in \mathbb{F}$*, and all* $\pi \leftarrow \mathsf{Open}(\mathsf{ck}, f, \alpha, \beta)$ *it holds that* $\mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi, \alpha, \beta) = 1$*.*

Definition 4 (Binding). *A polynomial commitment scheme* $\mathsf{PC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *over* $\mathbb{F}$ *is binding, if for all* $\lambda, d \in \mathbb{N}$ *and all PPT adversaries* $\mathcal{A}$ *it holds that*

$$\Pr\left[\begin{array}{r} \beta_0 \neq \beta_1 \\ \wedge \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi_0, \alpha, \beta_0) = 1 \\ \wedge \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi_1, \alpha, \beta_1) = 1 \end{array} : \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (\mathtt{com}, \alpha, \beta_0, \beta_1, \pi_0, \pi_1) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

2.4 KZG Commitments

Let us recall the KZG polynomial commitment scheme. The scheme's public parameters are powers of a secret point τ in the exponent of a group generator. Committing to a polynomial is done by evaluating it in the exponent at point τ, which can be done using the public parameters, but without knowledge of τ.

```
$\mathsf{Setup}(1^\lambda, 1^d)$
    $\tau \leftarrow \mathbb{F}_p$
    return $([\tau^0]_1, \ldots, [\tau^d]_1, [1]_2, [\tau]_2)$

$\mathsf{Commit}(\mathsf{ck}, f)$
    $\mathtt{com} := \sum_{i=0}^{d} f_i \cdot [\tau^i]_1$
    return $\mathtt{com}$

$\mathsf{Open}(\mathsf{ck}, \mathtt{com}, f, \alpha, \beta)$
    $q(X) := \frac{f(X) - \beta}{X - \alpha}$
    $\pi := \sum_{i=0}^{d} q_i \cdot [\tau^i]_1$
    return $\pi$

$\mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi, \alpha, \beta)$
    return $e(\mathtt{com} - [\beta]_1, [1]_2) \stackrel{?}{=} e(\pi, [\tau]_2 - [\alpha]_2)$
```

Fig. 1. The KZG polynomial commitment scheme.

Theorem 1 ([9]). *If the d-DLOG assumption holds with respect to* GGen*, then KZG commitment scheme described in Fig. 1 is a correct and binding polynomial commitment scheme in the AGM.*

It was shown by Feist and Khovratovich [14] that it is possible to open a KZG commitment in n positions much more efficiently than naively computing each opening separately Specifically, it is possible to perform a batch opening at n points in time $\mathcal{O}(n \log^2(n))$. This can be further improved to $\mathcal{O}(n \log(n))$, if the points are powers of a root of unity. We refer to this batch opening algorithm as $\mathsf{BatchOpen}$.

2.5 Weakly-Hiding Vector Commitments

A vector commitment allows for computing a short value $\texttt{com}$ for a potentially long vector of messages $(m_1, \ldots, m_n)$. Later on, one can compute short openings that certify that the vector committed to by $\texttt{com}$ opens to m_i at some position $i \in [n]$. Vector commitment should be binding in the usual sense, meaning that no position can be opened to two different values. In addition, we will require a weak form of hiding from our vector commitments.

Definition 5. *A vector commitment with batch opening is a tuple of PPT algorithms* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{BatchOpen}, \mathsf{Verify})$ *defined as follows:*

$\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$**:** *The setup algorithm takes security parameter λ and vector length n as input and returns commitment key* ck*.*

$(\texttt{com}, \texttt{aux}) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{m})$**:** *The commitment algorithm takes commitment key* ck *and vector $\boldsymbol{m} \in \mathcal{M}^n$ as input and returns commitment* $\texttt{com}$ *and auxiliary output* $\texttt{aux}$*.*

$(\pi_1, \ldots, \pi_n) \leftarrow \mathsf{BatchOpen}(\mathsf{ck}, \texttt{com}, \texttt{aux})$**:** *The batch opening algorithm takes commitment key* ck*, a commitment* $\texttt{com}$*, and auxiliary input* $\texttt{aux}$*, as input and returns openings $\pi_1, \ldots, \pi_n$.*

$b \leftarrow \mathsf{Verify}(\mathsf{ck}, \texttt{com}, \pi, i, m)$*:] The verification algorithm takes commitment key* ck*, commitment* $\texttt{com}$*, opening π, index i and message m as input and returns a bit b.*

Definition 6 (Correctness). *A vector commitment* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{BatchOpen}, \mathsf{Verify})$ *is correct, if for all $\lambda, n \in \mathbb{N}$, all $\boldsymbol{m} \in \mathcal{M}^n$, all* $(\texttt{com}, \texttt{aux}) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{m})$*, all* $(\pi_1, \ldots, \pi_n) \leftarrow \mathsf{BatchOpen}(\mathsf{ck}, \texttt{com}, \texttt{aux})$*, and all $i \in [n]$ it holds that* $\mathsf{Verify}(\mathsf{ck}, \texttt{com}, \pi_i, i, m) = 1$*.*

Definition 7 (Position-Binding). *A vector commitment* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{BatchOpen}, \mathsf{Verify})$ *is position binding if for all $\lambda, n \in \mathbb{N}$ and all PPT adversaries $\mathcal{A}$ it holds that*

$$\Pr\left[\begin{array}{r} m_0 \neq m_1 \\ \wedge \mathsf{Verify}(\mathsf{ck}, \texttt{com}, \pi_0, i, m_0) = 1 : \\ \wedge \mathsf{Verify}(\mathsf{ck}, \texttt{com}, \pi_1, i, m_1) = 1 \end{array} \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (\texttt{com}, i, m_0, m_1, \pi_0, \pi_1) \leftarrow \mathcal{A}(\mathsf{ck}) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 8 (Perfect Weak Hiding). *A vector commitment* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{BatchOpen}, \mathsf{Verify})$ *is perfectly weakly hiding if for all $\lambda, n \in \mathbb{N}$ and all $\boldsymbol{m}_0, \boldsymbol{m}_1 \in \mathcal{M}^n$ and all* $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$ *it holds that over the random coins of the commitment algorithm,* $\texttt{com}_0$ *and* $\texttt{com}_1$ *computed as* $(\texttt{com}_0, \texttt{aux}_0) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{m}_0)$ *and* $(\texttt{com}_1, \texttt{aux}_1) \leftarrow \mathsf{Commit}(\mathsf{ck}, \boldsymbol{m}_1)$ *are distributed identically.*

Definition 9 (Efficiency). *A vector commitment* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{BatchOpen}, \mathsf{Verify})$ *is efficient, if commitments are of size independent of n, the commitment algorithm runs in time $n \cdot \mathsf{poly}(\lambda)(\lambda)$ and the batch opening algorithm runs in time $n \cdot \mathsf{poly}(\lambda)(\log n, \lambda)$.*

```
Setup(1^λ, 1^n)
  pp ← KZG.Setup(1^λ, 1^n)
  return pp

Commit(ck, m)
  $r \leftarrow \mathbb{F}_p$
  $\mathbb{F}_p[X] \ni f(X) := \begin{cases} m_i \text{ if } X \in [n] \\ r \text{ if } X = 0 \end{cases}$
  com ← KZG.Commit(pp, f)
  return (com, f)

BatchOpen(ck, com, f)
  $\boldsymbol{\pi} \leftarrow$ KZG.BatchOpen(ck, com, f, $(i, f(i))_{i \in [n]}$)
  return $\boldsymbol{\pi}$

Verify(ck, com, π, i, m)
  return KZG.Verify(ck, com, π, i, m)
```

Fig. 2. Weakly-hiding vector commitments from KZG polynomial commitments.

Given a polynomial commitment, it is easy to construct a vector commitment by encoding the message vector's entries into distinct polynomial evaluations. In the theorem statement below, we recall this transformation using KZG commitments but add a small twist that makes the construction weakly hiding.

Theorem 2. *If* KZG *is a binding polynomial commitment scheme, then the construction specified in Fig. 2 is a correct, efficient, computationally position binding, and perfectly weakly hiding commitment scheme.*

Proof. Correctness is immediate from the correctness of KZG. Efficiency similarly follows from the definition of KZG and the batch opening algorithm of Feist and Khovratovich [14].

It is trivial to see, that any adversary breaking position binding of the vector commitment immediately also breaks binding of the KZG commitment scheme with the same probability. Position binding therefore follows immediately from the binding property of the KZG commitment. To see that the scheme is perfectly weakly hiding, consider any $\lambda, n \in \mathbb{N}$ and any $\boldsymbol{m}_0, \boldsymbol{m}_1 \in \mathcal{M}^n$ and any $\mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$. Any commitment to either vector has the form $\mathtt{com}_b = [\mu_b]_1 + [r]_1$, where μ_b is defined by m_b and independent of r. Since r is chosen uniformly from $\mathbb{F}_p$, $\mathtt{com}_b$ is always a uniformly distributed element of $\mathbb{G}_1$. In particular, this means that $\mathtt{com}_0$ and $\mathtt{com}_1$ are distributed identically.

2.6 Symmetric Encryption

We quickly recall the definition of a symmetric encryption scheme.

Definition 10. *A symmetric encryption scheme with keyspace $\mathcal{K}$ and message space $\mathcal{M}$ is a pair of PPT algorithms* $\mathsf{SE} = (\mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ *defined as follows:*

$\mathtt{ct} \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathcal{K}, m)$**:** *The setup algorithm takes a key* $\mathtt{k} \in \mathcal{K}$ *and a message* $m \in \mathcal{M}$ *as input and returns ciphertext* $\mathtt{ct}$.

$\mathtt{ct} \leftarrow \mathsf{Dec}^{\mathsf{sym}}(\mathcal{K}, \mathtt{ct})$**:** *The setup algorithm takes a key* $\mathtt{k} \in \mathcal{K}$ *and a ciphertext* $\mathtt{ct}$ *as input and returns message* m.

Definition 11 (Correctness). *A symmetric encryption scheme* $\mathsf{SE} = (\mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ *is correct, if for all* $\mathtt{k} \in \mathcal{K}$, *all* $m \in \mathcal{M}$, *and all* $\mathtt{ct} \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathtt{k}, m)$ *it holds that* $\mathsf{Dec}^{\mathsf{sym}}(\mathtt{k}, \mathtt{ct}) = m$.

We will only require a very weak notion of security, namely indistinguishability against eavesdroppers, also called EAV-security. This security notion is e.g. satisfied by the one-time pad or a (nonce-less) stream cipher.

Definition 12 (EAV-Security). *A symmetric encryption scheme* $\mathsf{SE} = (\mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ *has indistinguishable encryptions in the presence of an eavesdropper, or is EAV-secure, if for any PPT adversary* $\mathcal{A}$ *it holds that*

$$\Pr[\mathsf{Expt}^{\mathsf{EAV}}_{\mathsf{SE},\mathcal{A}}(1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

where the experiment is defined as follows.

```
$\underline{\mathsf{Expt}^{\mathsf{EAV}}_{\mathsf{SE},\mathcal{A}}(1^\lambda)}$
$\mathtt{k} \leftarrow \mathcal{K}$
$(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda)$
$b \leftarrow \{0,1\}$
$\mathtt{ct} \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathtt{k}, m_b)$
$b' \leftarrow \mathcal{A}(\mathtt{ct})$
return $\begin{cases} 1 & \text{if } b' = b \\ 0 & \text{otherwise} \end{cases}$
```

3 Extractable Witness KEMs

As a building block for our main constructions, we first define and instantiate the notion of an extractable witness KEM. This notion, with minor differences, has previously been defined by Choi and Vaudenay [11]. In their work, the authors present an instantiation of this notion for some class of problems using a new non-falsifiable hardness assumption in a new restricted non-standard computational model. In our work, we will focus on instantiating the notion of an extractable witness KEM for specific relations using a standard assumption in the AGM.

We first define the notions of an indexed family of NP relations. We will later define extractable witness KEMs relative to such a family. This is more general than defining them for an individual relation and allows some part of the relation to be fixed by system parameters.

Definition 13. *Let* $\mathcal{I} \subseteq \{0,1\}^*$ *be a set. A set* $\mathcal{F} = \{\mathcal{R}_I\}_{I \in \mathcal{I}}$ *a family of NP relations with index set* $\mathcal{I}$ *if for all* $I \in \mathcal{I}$, $\mathcal{R}_I$ *is an NP relation. We call* I *the index of* $\mathcal{R}_I$ *and* $\mathcal{R}_I$ *the relation identified by* I. *We use* $\mathcal{L}_I$ *to refer to the corresponding NP language.*

Relative to these families we can now define a witness KEM. The general idea of a witness KEM is that it works like a regular key encapsulation mechanism, but uses the pairs of statement and witness in an NP relation as a key-pair.

Definition 14. *A witness key encapsulation mechanism for a family of NP relations $\mathcal{F}$ and a keyspace $\mathcal{K}$ is a pair of PPT algorithms* $\mathsf{WKEM} = (\mathsf{Encap}, \mathsf{Decap})$*, defined as follows:*

$(\mathtt{ct}, \mathtt{k}) \leftarrow \mathsf{Encap}(I, \mathtt{x})$**:** *The encapsulation algorithm takes as input an index I identifying a relation $\mathcal{R}_I \in \mathcal{F}$ and a statement* $\mathtt{x}$ *and returns as output a ciphertext* $\mathtt{ct}$ *and a key* $\mathtt{k} \in \mathcal{K}$.

$\mathtt{k} \leftarrow \mathsf{Decap}(I, \mathtt{w}, \mathtt{ct})$**:** *The deterministic decapsulation algorithm takes as input an index I identifying a relation $\mathcal{R}_I \in \mathcal{F}$, a witness* $\mathtt{w}$*, and a ciphertext* $\mathtt{ct}$ *and returns a key* $\mathtt{k} \in \mathcal{K}$.

Definition 15 (Correctness). *A witness KEM* $\mathsf{WKEM} = (\mathsf{Encap}, \mathsf{Decap})$ *for a family of NP relations $\mathcal{F}$ is correct, if for any $\mathcal{R}_I \in \mathcal{F}$, any $\lambda \in \mathbb{N}$, any $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_I$, and any $(\mathtt{ct}, \mathtt{k}) \leftarrow \mathsf{Encap}(I, \mathtt{x})$ it holds that* $\mathsf{Decap}(I, \mathtt{w}, \mathtt{ct}) = \mathtt{k}$.

The above definitions by themselves do not guarantee any kind of security. Security will be derived from the extractability of the scheme. Extractability essentially says that if any efficient adversary can distinguish between a key encapsulated under some statement $\mathtt{x}$ and a random key, then this adversary can also be used to extract a witness $\mathtt{w}$ for $\mathtt{x}$.

Definition 16 (Extractability). *A witness KEM* $\mathsf{WKEM} = (\mathsf{Encap}, \mathsf{Decap})$ *for a family of NP-relations $\mathcal{F}$ is extractable, if there exists a PPT algorithm* Ext *such that for any stateful PPT adversary $\mathcal{A}$ and any relation $\mathcal{R}_I \in \mathcal{F}$ such that*

$$\Pr[\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{A}}(1^\lambda, I) = 1] \geq \frac{1}{2} + \epsilon(\lambda)$$

for some non-negligible function $\epsilon(\lambda)$, it holds that

$$\Pr\left[(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}} : \begin{array}{r} \mathtt{x} \leftarrow \mathcal{A}(1^\lambda, I) \\ \mathtt{w} \leftarrow \mathsf{Ext}^{\mathcal{A}(\cdot,\cdot)}(I, \mathtt{x}) \end{array}\right] \geq \delta(\lambda),$$

for some non-negligible function $\delta(\lambda)$. The latter probability is taken over the random coins of the adversary and the extractor and the experiment $\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{A}}(1^\lambda)$ *is defined as follows.*

```
$\underline{\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{A}}(1^\lambda, I)}$
$\mathtt{x} \leftarrow \mathcal{A}(1^\lambda, I)$
$b \leftarrow \{0,1\}$
$(\mathtt{ct}, \mathtt{k}_0) \leftarrow \mathsf{Encap}(I, \mathtt{x})$
$\mathtt{k}_1 \leftarrow \mathcal{K}$
$b' \leftarrow \mathcal{A}(\mathtt{ct}, \mathtt{k}_b)$
$\mathbf{return} \begin{cases} 1 & \textit{if } b' = b \\ 0 & \textit{otherwise} \end{cases}$
```

3.1 An Extractable Witness KEM for KZG Openings

We now proceed with constructing an extractable witness KEM as just defined for a very specific family of relations, specifically those describing valid opening of KZG commitments. Let $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ be the KZG commitment relative to the bilinear group $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p as specified in Fig. 1. Let $\mathsf{CK} = \{\mathsf{ck} \in \mathbb{G}_1^{d+1} \times \mathbb{G}_2^2 \mid d \in \mathbb{N} \wedge \mathsf{ck} \in \mathsf{Setup}(1^\lambda, 1^d)\}$ be the set of valid KZG commitment keys. We can then define the family of NP relations of valid KZG openings as

$$\mathcal{F}_{KZG} := \{\mathcal{R}_{\mathsf{ck}}\}_{\mathsf{ck} \in \mathsf{CK}}$$

where

$$\mathcal{R}_{\mathsf{ck}} = \left\{ \left((\mathtt{com}_j, \alpha_j, \beta_j)_{j \in [\ell]}, (\pi_j)_{j \in [\ell]} \right) \mid \forall j \in [\ell].\ \mathsf{Verify}(\mathsf{ck}, \mathtt{com}_j, \pi_j, \alpha_j, \beta_j) = 1 \right\}$$

for any $\mathsf{ck} \in \mathsf{CK}$.

```
$\mathsf{Encap}^{\mathsf{H}}(\mathsf{ck}, (\mathtt{com}, \alpha, \beta))$

for $1 \le j \le \ell$
    $r_j \leftarrow \mathbb{F}_p$
    $s_j := e(r_j \cdot (\mathtt{com} - [\beta_j]_1), [1]_2)$
    $\mathtt{ct}_j \leftarrow r_j \cdot ([\tau]_2 - [\alpha_j]_2)$
$\mathtt{ct} := (\mathtt{ct}_1, \dots, \mathtt{ct}_\ell)$
$\mathtt{k} := \mathsf{H}(s_1, \dots, s_\ell)$
return $(\mathtt{ct}, \mathtt{k})$
```

```
$\mathsf{Decap}^{\mathsf{H}}(\mathsf{ck}, (\pi_1, \dots, \pi_\ell), \mathtt{ct})$

parse $\mathtt{ct}$ as $(\mathtt{ct}_1, \dots, \mathtt{ct}_\ell)$
for $1 \le j \le \ell$
    $s_j := e(\pi_j, \mathtt{ct}_j)$
$\mathtt{k} := \mathsf{H}(s_1, \dots, s_\ell)$
return $\mathtt{k}$
```

Fig. 3. Construction of an Extractable Witness KEM for $\mathcal{F}_{KZG}$ with keyspace $\{0,1\}^\lambda$ in the combined Algebraic Group and Random Oracle Model.

Theorem 3. *Let $\mathsf{H} : \mathbb{G}_T^* \rightarrow \{0,1\}^\lambda$ be a hash functioned modeled as a random oracle. If the d-DLOG assumption holds with respect to* GGen*, then the construction described in Fig. 3 is an extractable witness KEM for $\mathcal{F}_{KZG}$ in the algebraic group model.*

Proof. Let $\mathcal{A}$ be an arbitrary algebraic PPT adversary with non-negligible advantage $\epsilon(\lambda)$ for the extractability of the construction described in Fig. 3.

We construct an extractor Ext as follows. The extractor receives as input an index $\mathsf{ck} = ([\tau^0]_1, \dots, [\tau^d]_1, [1]_2, [\tau]_2) \in \mathsf{CK}$ as well as a statement $((\mathtt{com}_1, \alpha_1, \beta_1), \dots, (\mathtt{com}_\ell, \alpha_\ell, \beta_\ell) \in (\mathbb{G}_1 \times \mathbb{F}_p^2)^\ell$. Further, since $\mathcal{A}$ is an algebraic algorithm, the extractor also receives an algebraic representations of each $\mathtt{com}_j$ in the form of coefficients $f_{j,0}, \dots, f_{j,d}$ such that

$$\mathtt{com}_j := \sum_{i=0}^{d} f_{j,i} \cdot [\tau^i]_1.$$

The extractor chooses $r_1, \dots, r_\ell \leftarrow \mathbb{F}_p$, computes $\mathtt{ct}_j := r_j \cdot ([\tau]_2 - [\alpha_j]_2)$ for each $j \in [\ell]$, sets $\mathtt{ct} := (\mathtt{ct}_1, \dots, \mathtt{ct}_\ell)$, chooses a random key $\mathtt{k} \leftarrow \{0,1\}^\lambda$, initializes an empty list $\Gamma := \emptyset$ and invokes $\mathcal{A}(\mathtt{ct}, \mathtt{k})$. The adversary $\mathcal{A}$ expects access to a random oracle that Ext simulates as follows. For any query $\boldsymbol{s} \in \mathbb{G}_T^*$ such that either $|\boldsymbol{s}| \neq \ell$ or for at least one $j \in [\ell]$, $s_j \neq e(r_j \cdot (\mathtt{com} - [\beta_j]_1), [1]_2)$, Ext continues to simulate the random oracle via lazy sampling. That is, if there exists an entry $(\boldsymbol{s}, \mathtt{k}_{\boldsymbol{s}}) \in \Gamma$ it returns $\mathtt{k}_{\boldsymbol{s}}$. If such an entry does not yet exist, Ext samples $\mathtt{k}_{\boldsymbol{s}} \leftarrow \{0,1\}^n$, adds $(\boldsymbol{s}, \mathtt{k}_{\boldsymbol{s}})$ to Γ, and returns $\mathtt{k}_{\boldsymbol{s}}$.

If, however, $|\boldsymbol{s}| = \ell$ and for all $j \in [\ell]$, $s_j = e(r_j \cdot (\mathtt{com} - [\beta_j]_1), [1]_2)$, then the extractor aborts $\mathcal{A}$ and continues as follows. Since $\mathcal{A}$ is an algebraic algorithm it also provides an algebraic description of each s_j in the form of of coefficients[3]

$$\tilde{w}_{j,0}, \dots, \tilde{w}_{j,2d}, \tilde{q}_{j,1,0}, \dots, \tilde{q}_{j,1,d}, \tilde{q}_{j,2,0}, \dots, \tilde{q}_{j,\ell,d}, \tilde{h}_{j,1}, \dots, \tilde{h}_{j,\ell}$$

such that

$$\begin{aligned} s_j :=& \sum_{i=0}^{2d} \tilde{w}_{j,i} \cdot [\tau^i]_T + \sum_{k=1}^{\ell} \sum_{i=0}^{d} \tilde{q}_{j,k,i} \cdot [r_k \cdot (\tau - \alpha_k) \cdot \tau^i]_T \\ &+ \sum_{k=1}^{\ell} \tilde{h}_{j,k} \cdot [r_j \cdot (\tau - \alpha_j) \cdot r_k \cdot (\tau - \alpha_k)]_T. \end{aligned}$$

Since Ext computed each $\mathtt{ct}_k$, for $k \neq j$ algebraically, however, we can simplify this representation. Specifically Ext can compute an algebraic description of each s_j in the form of coefficients $w_{j,0}, \dots, w_{j,2d}, q_{j,0}, \dots, q_{j,d}, h_j$ defined as

$$w_{j,i} := \begin{cases} \tilde{w}_{j,i} + \sum_{k \in [\ell] \setminus \{j\}} \tilde{q}_{j,k,i} r_k \alpha_k & \text{if } i = 0 \\ \tilde{w}_{j,i} + \sum_{k \in [\ell] \setminus \{j\}} \tilde{q}_{j,k,i} r_k \alpha_k + \tilde{q}_{j,k,i-1} r_k & \text{if } 0 < i \leq d \\ \tilde{w}_{j,i} + \sum_{k \in [\ell] \setminus \{j\}} \tilde{q}_{j,k,i-1} r_k & \text{if } i = d+1 \\ \tilde{w}_{j,i} & \text{if } i > d+1 \end{cases}$$

$$q_{j,i} := \begin{cases} \tilde{q}_{j,j,i} + \sum_{k \in [\ell] \setminus \{j\}} \tilde{h}_{j,k} r_k & \text{if } i = 0 \\ \tilde{q}_{j,j,i} + \sum_{k \in [\ell] \setminus \{j\}} \tilde{h}_{j,k} r_k \alpha_k & \text{if } i = 1 \\ \tilde{q}_{j,j,i} & \text{if } i > 1 \end{cases}$$

$$h_j := \tilde{h}_{j,j}$$

such that

$$\begin{aligned} s_j :=& \sum_{i=0}^{2d} w_{j,i} \cdot [\tau^i]_T + \sum_{i=0}^{d} q_{j,i} \cdot [r_j \cdot (\tau - \alpha_j) \cdot \tau^i]_T + h_j \cdot [r^2 \cdot (\tau - \alpha_j)^2)]_T \\ =& \sum_{i=0}^{2d} w_{j,i} \cdot [\tau^i]_T + r_j \cdot (\tau - \alpha_j) \cdot \sum_{i=0}^{d} q_{j,i} \cdot [\tau^i]_T + h_j r_j^2 \cdot [(\tau - \alpha_j)^2]_T \end{aligned}$$

[3] Note that we allow the algebraic adversary the maximum amount of freedom here that they would only have in the context of a Type-1 pairing. In the context of Type-2 or Type-3 pairings, many of these coefficients would necessarily be 0.

For each $j \in [\ell]$ we define the two polynomials

$$F_j(X) := \sum_{i=1}^{d} f_{j,i} \cdot X^i$$

and

$$G_j(X,Y) := \sum_{i=0}^{2d} w_{j,i} \cdot X^i + Y \cdot (X - \alpha_j) \cdot \sum_{i=0}^{d} q_{j,i} \cdot X^i + h_j Y^2 \cdot (X - \alpha_j)^2.$$

It is easy to see that $\texttt{com}_j = [F_j(\tau)]_T$ and $s_j = [G_j(\tau, r_j)]_T$.

If for all $j \in [\ell]$, $G_j(X,Y) = Y(F_j(X) - \beta_j)$ the extractor computes $\pi_j := \sum_{i=1}^{d} q_{j,i} \cdot [\tau^i]_1$ and outputs $\boldsymbol{\pi} := (\pi_1, \ldots, \pi_\ell)$. Otherwise the extractor outputs $\perp$. If the adversary terminates without querying an $\boldsymbol{s}$ as described above, then the extractor also outputs $\perp$.

We now prove two claims about the extractor.

Claim 4. *If* Ext *outputs* $\boldsymbol{\pi} \neq \perp$*, then for all* $j \in [\ell]$ *it holds that* $\mathsf{Verify}(\mathsf{ck}, \texttt{com}_j, \pi_j, \alpha_j, \beta_j) = 1$.

Proof. The extractor only outputs $\boldsymbol{\pi} \neq \perp$, if for all $j \in [\ell]$, $G_j(X,Y) = Y(F_j(X) - \beta_j)$. We observe that, this can only be true if $h_j = 0$ and $w_{j,i} = 0$ for all $j \in [\ell]$ and $0 \leq i \leq 2d$. Therefore,

$$G_j(X,Y) := Y \cdot (X - \alpha_j) \cdot \sum_{i=0}^{d} q_{j,i} \cdot X^i.$$

and thus

$$\begin{aligned} & G_j(X,Y) = Y(F_j(X) - \beta_j) \\ \iff & Y \cdot (X - \alpha_j) \cdot \sum_{i=0}^{d} q_{j,i} \cdot X^i = Y(F_j(X) - \beta_j) \\ \iff & \sum_{i=0}^{d} q_{j,i} \cdot X^i = \frac{F_j(X) - \beta_j}{X - \alpha_j}. \end{aligned}$$

It follows that the extractor's output

$$\pi_j = \left[\frac{F_j(\tau) - \beta_j}{\tau - \alpha_j}\right]_1$$

and thus the verification equation

$$\begin{aligned} e(\pi_j, [\tau]_2 - [\alpha_j]_2) &= e\left(\left[\frac{F_j(\tau) - \beta_j}{\tau - \alpha_j}\right]_1, [\tau - \alpha_j]_2\right) \\ &= [F_j(\tau) - \beta_j]_T \\ &= e([F_j(\tau) - \beta_j]_1, [1]_2) = e(\texttt{com}_j - [\beta_j]_1, [1]_2) \end{aligned}$$

holds as required. □

Claim 5. Ext *outputs* $\boldsymbol{\pi} = \bot$ *with probability at most* $1 - 2\epsilon(\lambda) + \mathsf{negl}(\lambda)$.

Proof. Let Hit denote the event that $\boldsymbol{s}$ with $|\boldsymbol{s}| = \ell$ and $s_j = e(r_j \cdot (\mathtt{com}_j - [\beta_j]_1), [1]_2)$ for all $j \in [\ell]$ is queried to the random oracle. The extractor outputs $\bot$, either if Hit does not occur, or if Hit occurs but for at least one $j \in [\ell]$ $G_j(X, Y) \neq Y(F_j(X) - \beta_j)$. Since those two events are mutually exclusive, we thus have

$$\Pr[\boldsymbol{\pi} = \bot] = \Pr[\overline{\mathsf{Hit}}] + \Pr[\mathsf{Hit} \wedge \exists j \in [\ell].\ G_j(X, Y) \neq Y(F_j(X) - \beta_j)]. \qquad (1)$$

We bound the two probabilities separately.

Let us first bound $\Pr[\overline{\mathsf{Hit}}]$. Recall, that $\mathcal{A}$ is an adversary with advantage $\epsilon(\lambda)$. Whenever Hit does not occur, the view of $\mathcal{A}$ remains independent of the bit b in the KEM-CPA experiment. This is the case, since unless Hit occurs, both $\mathtt{k}_0$ and $\mathtt{k}_1$ are uniformly distributed. Therefore we have that

$$\begin{aligned}
\frac{1}{2} + \epsilon(\lambda) &= \Pr[\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM}_{KZG},\mathcal{A}}(1^\lambda) = 1] \\
&= \Pr[b' = b] \\
&= \overbrace{\Pr[b' = b \mid \mathsf{Hit}]}^{\leq 1} \Pr[\mathsf{Hit}] + \overbrace{\Pr[b' = b \mid \overline{\mathsf{Hit}}]}^{=1/2} \Pr[\overline{\mathsf{Hit}}] \\
&\leq 1 - \Pr[\overline{\mathsf{Hit}}] + \frac{1}{2} \Pr[\overline{\mathsf{Hit}}] \\
&= 1 - \frac{1}{2} \Pr[\overline{\mathsf{Hit}}]
\end{aligned}$$

and thus, we have that

$$\Pr[\overline{\mathsf{Hit}}] \leq 1 - 2\epsilon(\lambda). \qquad (2)$$

Next we bound $\Pr[\mathsf{Hit} \wedge \exists j \in [\ell].\ G_j(X, Y) \neq Y(F_j(X) - \beta_j)]$. For each $j \in [\ell]$, define the bivariate polynomial $Q_j(X, Y) = G_j(X, Y) - Y(F_j(X) - \beta_j)$. Since Hit happened, it must hold that $s_j = e(r_j \cdot (\mathtt{com}_j - [\beta_j]_1), [1]_2) = [r_j \cdot (F_j(\tau) - \beta_j)]_T$ and since $s_j = [G_j(\tau, r_j)]_T$, it must hold that $Q_j(\tau, r_j) = 0$. However, since $G_j(X, Y) \neq Y(F_j(X) - \beta_j)$, it also holds that $Q_j(X, Y) \neq 0$, meaning $Q_j(X, Y)$ is a non-zero polynomial with a root at (τ, r_j).

We can rewrite $Q_j(X, Y)$ as a polynomial of the form $Q_j(X, Y) = C_{j,0}(X) + C_{j,1}(X) \cdot Y + C_{j,2}(X) \cdot Y^2$, where each $C_{j,i}(X) \in \mathbb{F}_p[X]$ is a univariate polynomial of degree at most $2d$. Since $Q_j(X, Y)$ is non-zero, at least one of $C_{j,0}(X), C_{j,1}(X)$, $C_{j,2}(X)$ must also be non-zero.

Now consider the univariate Polynomial $P_j(Y) := Q_j(\tau, Y)$. We can consider two cases, either $P_j(Y) = 0$, or $P_j(Y) \neq 0$. In the case that $P_j(Y) = 0$, it would need to hold that $C_{j,0}(\tau) = C_{j,1}(\tau) = C_{j,2}(\tau) = 0$, but at least one of these is a non-zero polynomial. We can therefore find the roots of one of the non-zero $C_{j,0}(Y), C_{j,1}(Y), C_{j,2}(Y)$ to recover τ. On the other hand, if $P_j(Y) \neq 0$, we can find the roots of $P_j(Y)$ to recover r_j.

We use this to specify the following reduction to d-DLOG. On input $[\eta^0]_1, \ldots, [\eta^d]_1, [\eta^0]_2, [\eta^1]_2$, the reduction $\mathcal{B}$ flips a bit $b \leftarrow \{0, 1\}$. If $b = 0$, then $\mathcal{B}$

defines $\mathsf{ck} = ([\eta^0]_1, \dots, [\eta^d]_1, [\eta^0]_2, [\eta^1]_2)$. If $b = 1$, then $\mathcal{B}$ chooses $\tau \leftarrow \mathbb{F}_q$ and defines $\mathsf{ck} = ([\tau^0]_1, \dots, [\tau^d]_1, [1]_2, [\tau]_2)$. In both cases, $\mathcal{B}$ then invokes $\mathcal{A}(1^\lambda, \mathsf{ck})$. Eventually $\mathcal{A}$ will output $((\mathtt{com}_1, \alpha_1, \beta_1), \dots, (\mathtt{com}_\ell, \alpha_\ell, \beta_\ell))$ together with the algebraic explanation of each $\mathtt{com}_j$. If $b = 0$, then for $j \in [\ell]$, the reduction chooses $r_j \leftarrow \mathbb{F}_p$ and computes $\mathtt{ct}_j := r_j \cdot ([\eta]_2 - [\alpha_j]_2)$. If $b = 1$, then for $j \in [\ell]$, the reduction chooses $z_j \leftarrow \mathbb{F}_p$ and computes $\mathtt{ct}_j := (\tau - \alpha_j) \cdot ([\eta]_2 + z_j \cdot [1]_2)$. Note, that, if we define $r_j := z_j + \eta$ it holds that $(\tau - \alpha_j) \cdot ([\eta]_2 + z_j \cdot [1]_2) = (z_j + \eta) \cdot ([\tau]_2 - [\alpha_j]_2) = r_j \cdot ([\tau]_2 - [\alpha_j]_2)$. Therefore, in the case of $b = 1$ the ciphertexts are computed correctly for implicitly defined *but uniformly distributed* r_j. From this point, $\mathcal{B}$ proceeds exactly as Ext until Hit occurs. It is important to verify that $\mathcal{B}$ can actually check whether this query occurs, even in the case $b = 1$. However, due to the bilinearity of the pairing, $\mathcal{B}$ can compute the relevant values as

$$\begin{aligned} e(\mathtt{com}_j - [\beta_j]_1, z_j \cdot [1]_2 + [\eta]_2) &= e(\mathtt{com}_j - [\beta_j]_1, (z_j + \eta) \cdot [1]_2) \\ = e(\mathtt{com}_j - [\beta_j]_1, r_j \cdot [1]_2) &= e(r_j \cdot (\mathtt{com}_j - [\beta_j]_1), [1]_2) \end{aligned}$$

and thus can check each query against these values. If Hit does not occur or if for all $j \in [\ell]$, it holds that $G_j(X, Y) = Y(F_j(X) - \beta_j)$, then $\mathcal{B}$ aborts. Otherwise, we consider two cases.

If $b = 0$ and there exists an index j^* such that $G_{j^*}(X, Y) \neq Y(F_{j^*}(X) - \beta_{j^*})$ and $P_{j^*}(Y) = 0$, then, as discussed above, there exists a $C_{j^*}(X) \in \{C_{j^*,0}(X), C_{j^*,1}(X), C_{j^*,2}(X)\}$ such that $C_{j^*}(X)$ is non-zero. $\mathcal{B}$ factors $C_{j^*}(X)$ to finds all of its at most $2d$ roots. For each root ξ, $\mathcal{B}$ checks whether $\xi \cdot [1]_1 = [\eta]_1$ and returns ξ if that's the case. Since η must be one of the roots, $\mathcal{B}$ will always correctly identify η in this case.

If $b = 1$ and there exists an index j^* such that $G_{j^*}(X, Y) = Y(F_{j^*}(X) - \beta_{j^*})$ and $P_{j^*}(Y) \neq 0$, then $\mathcal{B}$ factors $P_{j^*}(Y)$ to find the all of its at most 2 roots. For each root ξ, $\mathcal{B}$ checks whether $\xi \cdot [1]_1 = z_{j^*} \cdot [1]_1 + [\eta]_1$ and returns $\xi - z_{j^*}$ if that is the case. Since $r_{j^*} = z_{j^*} + \eta$ must be one of the roots, $\mathcal{B}$ will always correctly identify $\eta = r_{j^*} - z_{j^*}$ in this case. If neither of those two cases happens, then $\mathcal{B}$ also aborts.

Since the view of $\mathcal{A}$ is independent of the value of b and for each j^* such that $G_{j^*}(X, Y) \neq Y(F_{j^*}(X) - \beta_{j^*})$, it must either be the case that $P_{j^*}(Y) = 0$ or that $P_{j^*}(Y) \neq 0$, it follows under the d-DLOG assumption, that

$$\begin{aligned} &\mathsf{negl}(\lambda) \\ &\geq \Pr\left[\tau = \tau' : \begin{array}{r} \mathsf{par} \leftarrow \mathsf{GGen}(1^\lambda), \tau \leftarrow \mathbb{F}_p \\ \tau' \leftarrow \mathcal{B}(\mathsf{par}, ([\tau^0]_1, \dots, [\tau^d]_1, [1]_2, [\tau]_2)) \end{array}\right] \\ &= \Pr[b = 0] \cdot \Pr\big[\mathsf{Hit} \wedge \big(\exists j^* \in [\ell].\ G_{j^*}(X, Y) \neq Y(F_{j^*}(X) - \beta_{j^*}) \wedge P_{j^*}(Y) = 0\big)\big] \\ &\quad + \Pr[b = 1] \cdot \Pr\big[\mathsf{Hit} \wedge \big(\exists j^* \in [\ell].\ G_{j^*}(X, Y) \neq Y(F_{j^*}(X) - \beta_{j^*}) \wedge P_{j^*}(Y) \neq 0\big)\big] \\ &= \frac{1}{2} \cdot \left(\begin{array}{r} \Pr[\mathsf{Hit} \wedge (\exists j^* \in [\ell].\ G_{j^*}(X, Y) \neq Y(F_{j^*}(X) - \beta_{j^*})) \wedge P_{j^*}(Y) = 0)] \\ + \Pr[\mathsf{Hit} \wedge (\exists j^* \in [\ell].\ G_{j^*}(X, Y) \neq Y(F_{j^*}(X) - \beta_{j^*})) \wedge P_{j^*}(Y) \neq 0)] \end{array} \right) \\ &\geq \frac{1}{2} \Pr[\mathsf{Hit} \wedge \exists j^* \in [\ell].\ G_{j^*}(X, Y) \neq Y(F_{j^*}(X) - \beta_{j^*}))]. \end{aligned}$$

The claim thus follows. □

By combining the two claims we finally get

$$\Pr\left[\forall j \in [\ell].\mathsf{Verify}(\mathsf{ck}, \mathtt{com}_j, \pi_j, \alpha_j, \beta_j) = 1 : \begin{array}{r} (\mathtt{com}_j, \alpha_j, \beta_j)_{j\in[\ell]} \leftarrow \mathcal{A}(1^\lambda, \mathsf{ck}) \\ \boldsymbol{\pi} \leftarrow \mathsf{Ext}^{\mathcal{A}(\cdot,\cdot)}(\mathsf{ck}, (\mathtt{com}_j, \alpha_j, \beta_j)_{j\in[\ell]}) \end{array}\right]$$
$$= \Pr[\pi \neq \bot] = 1 - \Pr[\pi = \bot] = 2\epsilon(\lambda) - \mathsf{negl}(\lambda).$$

which is non-negligible for any non-negligible function $\epsilon(\lambda)$, as required. □

We will later require an extractable witness KEM not just for plain KZG, but in fact for the derived weakly hiding vector commitment described in Fig. 3. Let VC be the vector commitment and let $\mathsf{CK}_{\mathsf{VC}} = \{\mathsf{ck} \mid \mathsf{ck} \in \mathsf{VC.Setup}(1^\lambda, 1^n)\}$ be the set of valid commitment keys. We can then define the family of NP relations of valid VC openings as

$$\mathcal{F}_{\mathsf{VC}} := \{\mathcal{R}_{\mathsf{ck}}\}_{\mathsf{ck}\in\mathsf{CK}_{\mathsf{VC}}}$$

where

$$\mathcal{R}_{\mathsf{ck}} = \left\{\left((\mathtt{com}_j, i_j, m_j)_{j\in[\ell]}, (\pi_j)_{j\in[\ell]}\right) \mid \forall j \in [\ell].\mathsf{VC.Verify}(\mathsf{ck}, \mathtt{com}_j, \pi_j, i_j, m_j) = 1\right\}$$

for any $\mathsf{ck} \in \mathsf{CK}_{\mathsf{VC}}$.

Since VC commitments are simply KZG commitments, openings are KZG openings, and verification is KZG verification, it follows immediately that $\mathcal{F}_{\mathsf{VC}} = \mathcal{F}_{KZG}$ and we thus get the following corollary from Theorem 3.

Corollary 6. *Let $\mathsf{H} : \mathbb{G}_T^* \to \{0,1\}^\lambda$ be a hash functioned modeled as a random oracle. If the d-DLOG assumption holds with respect to* GGen*, then the construction described in Fig. 3 is an extractable witness KEM for $\mathcal{F}_{\mathsf{VC}}$ in the algebraic group model.*

4 Extractable Witness Encryption

In this section, we recall the definition of extractable witness encryption following broadly the definitions of [21], with the extension to *families* of relations. We then go on to show that one can generically construct it from any extractable witness KEM using the standard KEM/DEM paradigm.

As is the case for regular encryption and KEMs, witness encryption and witness KEMs are very similar, with the only difference being that a witness encryption scheme is capable of encrypting a freely chosen message instead of a random one. The definitions are otherwise very similar to the definitions from the previous section.

Definition 17. *A witness encryption scheme for a family of NP relations $\mathcal{F}$ and a messagespace $\mathcal{M}$ is a pair of PPT algorithms* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$*, defined as follows:*

$\mathtt{ct} \leftarrow \mathsf{Enc}(I, \mathtt{x}, m)$**:** *The encryption algorithm takes as input an index I identifying a relation $\mathcal{R}_I \in \mathcal{F}$, a statement* $\mathtt{x}$*, and a message $m \in \mathcal{M}$ and returns as output a ciphertext* $\mathtt{ct}$.

$m/\bot \leftarrow \mathsf{Dec}(I, \mathtt{w}, \mathtt{ct})$: *The deterministic decryption algorithm takes as input an index I identifying a relation $\mathcal{R}_I \in \mathcal{F}$, a ciphertext* $\mathtt{ct}$, *and a witness* $\mathtt{w}$ *and returns a message $m \in \mathcal{M}$ or a error symbol $\bot$.*

Definition 18 (Correctness). *A witness encryption scheme* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ *for a family of NP relations $\mathcal{F}$ is correct, if for any $\mathcal{R}_I \in \mathcal{F}$, any $\lambda \in \mathbb{N}$, any $(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_I$, any $m \in \mathcal{M}$, and any $\mathtt{ct} \leftarrow \mathsf{Enc}(\mathtt{x})$ it holds that* $\mathsf{Dec}(I, \mathtt{w}, \mathtt{ct}) = m$.

Definition 19 (Extractability). *A witness encryption scheme* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ *for a family of NP-relations $\mathcal{F}$ is extractable, if there exists a PPT algorithm* Ext *such that for any stateful PPT adversary $\mathcal{A}$ and any relation $\mathcal{R}_I \in \mathcal{F}$ such that*

$$\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda, I) = 1] \geq \frac{1}{2} + \epsilon(\lambda),$$

for some non-negligible function $\epsilon(\lambda)$ it holds that

$$\Pr\left[(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}} : \begin{array}{r} (\mathtt{x}, m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, I) \\ \mathtt{w} \leftarrow \mathsf{Ext}^{\mathcal{A}(\cdot)}(I, \mathtt{x}, m_0, m_1) \end{array}\right] \geq \delta(\lambda),$$

for some non-negligible function $\delta(\lambda)$. The latter probability is taken over the random coins of the adversary and the extractor and the experiment $\mathsf{Expt}^{\mathsf{CPA}}_{\mathcal{A}}$ *is defined as follows.*

```
$\underline{\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda, I)}$
$(\mathtt{x}, m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, I)$
$b \leftarrow \{0,1\}$
$\mathtt{ct} \leftarrow \mathsf{Enc}(I, \mathtt{x}, m_b)$
$b' \leftarrow \mathcal{A}(\mathtt{ct})$
$\textbf{return} \begin{cases} 1 & \textit{if } b' = b \\ 0 & \textit{otherwise} \end{cases}$
```

4.1 Extractable Witness Encryption from Extractable Witness KEMs

We can construct extractable witness encryption for a family of NP-relations $\mathcal{F}$ and a message space $\mathcal{M}$ from any extractable witness KEM for $\mathcal{F}$ and any EAV secure symmetric encryption scheme with message space $\mathcal{M}$, as long as the two schemes share a compatible key space $\mathcal{K}$. The construction, shown in Fig. 4 essentially follows the standard KEM/DEM paradigm instantiated with an extractable witness KEM. Even though the security of the KEM/DEM paradigm has been proven ad nauseam, we will not skip the proof here. Since we are not proving indistinguishability, but *extractability* we need to be a bit more careful. Note that encapsulated keys are in general *not* indistinguishable from random keys. In fact, since the *adversary* chooses the statement, they may very well *know the witness* and thus be capable of distiguishing with overwhelming probability.

```
$\mathsf{Enc}(I, \mathtt{x}, m)$
$(\mathtt{ct}_1, \mathtt{k}) \leftarrow \mathsf{Encap}(I, \mathtt{x})$
$\mathtt{ct}_2 \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathtt{k}, m)$
return $(\mathtt{ct}_1, \mathtt{ct}_2)$
```

```
$\mathsf{Dec}(I, \mathtt{w}, (\mathtt{ct}_1, \mathtt{ct}_2))$
$\mathtt{k} := \mathsf{Decap}(I, \mathtt{w}, \mathtt{ct}_1)$
$m := \mathsf{Dec}^{\mathsf{sym}}(\mathtt{k}, \mathtt{ct}_2)$
return $m$
```

Fig. 4. Construction of an extractable witness encryption Scheme for $\mathcal{F}$ based on an extractable witness KEM and an EAV secure symmetric encryption scheme.

It is merely the case in this case *we are capable of extracting the witness.* We need to be careful in our proof that this capability is preserved in the KEM/DEM paradigm. This may not always be the case, depending on how one executes the standard hybrid argument.

Theorem 7. *Let* $\mathsf{WKEM} = (\mathsf{Encap}, \mathsf{Decap})$ *be an extractable witness KEM for* $\mathcal{F}$ *and key space* $\mathcal{K}$. *Let* $\mathsf{SE} = (\mathsf{Enc}^{\mathsf{sym}}, \mathsf{Dec}^{\mathsf{sym}})$ *be an EAV secure symmetric encryption scheme with key space* $\mathcal{K}$, *message space* $\mathcal{M}$. *Then* $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ *as specified in Fig. 4 is an extractable witness encryption scheme for* $\mathcal{F}$ *and message space* $\mathcal{M}$.

Proof. We first define a *modified* witness encryption scheme $\widetilde{\mathsf{WE}} = (\widetilde{\mathsf{Enc}}, \cdot)$ that, instead of using the encapsulated key $\mathtt{k}$ to perform the symmetric encryption, chooses a fresh key $\mathtt{k}' \leftarrow \mathcal{K}$ to do so. This scheme has no well-defined decryption algorithm.Nevertheless, for any PPT adversary $\mathcal{A}$, the probability $\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\widetilde{\mathsf{WE}},\mathcal{A}}(1^\lambda, I) = 1]$ is still well defined. Let $\mathcal{A}$ be an arbitrary PPT adversary such that

$$\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda, I) = 1] = \frac{1}{2} + \epsilon(\lambda). \tag{3}$$

for some non-negligible function $\epsilon(\lambda)$. We first prove the following claim.

Claim 8. *For any* $I \in \mathcal{I}$, *it holds that*

$$\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\widetilde{\mathsf{WE}},\mathcal{A}}(1^\lambda, I) = 1] \leq \tfrac{1}{2} + \mathsf{negl}(\lambda).$$

Proof. We construct a PPT adversary $\mathcal{B}$ against the EAV security of SE as follows. Upon input 1^λ, $\mathcal{B}$ invokes $\mathcal{A}(1^\lambda, I)$, receiving $\mathtt{x}, m_0, m_1$ in response, and outputs m_0, m_1. After receiving as input the challenge ciphertext $\mathtt{ct}_2$, it compues $(\mathtt{ct}_1, \mathtt{k}) \leftarrow \mathsf{Encap}(I, \mathtt{x})$ and invokes $\mathcal{A}((\mathtt{ct}_1, \mathtt{ct}_2))$. In response, $\mathcal{A}$ will output a bit b', which $\mathcal{B}$ also outputs.

It is easy to see that $\mathcal{B}$ perfectly simulates the $\mathsf{Expt}^{\mathsf{CPA}}_{\widetilde{\mathsf{WE}},\mathcal{A}}(1^\lambda, I)$ experiment for $\mathcal{A}$. Further, whenever $\mathcal{A}$ would be successful, so is $\mathcal{B}$. It thus follows from the EAV security of SE that

$$\frac{1}{2} + \mathsf{negl}(\lambda) \geq \Pr[\mathsf{Expt}^{\mathsf{EAV}}_{\mathsf{SE},\mathcal{B}}(1^\lambda) = 1] = \Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\widetilde{\mathsf{WE}},\mathcal{A}}(1^\lambda, I) = 1]$$

as claimed. □

Claim 9. *There exists a PPT adversary $\mathcal{B}$ such that for any $I \in \mathcal{I}$, it holds that*

$$\Pr[\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{B}}(1^\lambda, I) = 1] = \tfrac{1}{2} + \tfrac{1}{2}\epsilon(\lambda) - \mathsf{negl}(\lambda).$$

Proof. Upon input 1^λ and I, $\mathcal{B}$ invokes $\mathcal{A}(1^\lambda, I)$, receiving $\mathtt{x}, m_0, m_1$ in response and outputs $\mathtt{x}$. After receiving $\mathtt{ct}_1, \mathtt{k}$, $\mathcal{B}$ samples $b' \leftarrow \{0,1\}$, computes $\mathtt{ct}_2 \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathtt{k}, m_{b'})$, and invokes $\mathcal{A}((\mathtt{ct}_1, \mathtt{ct}_2))$. Eventually $\mathcal{A}$ will output a bit b'' and $\mathcal{B}$ will output 0, if $b'' = b'$ and 1 otherwise.

Let b denote the random bit of the experiment $\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{B}}(1^\lambda, I)$. It is then easy to see that, if $b = 0$, then $\mathcal{B}$ perfectly simulates the experiment $\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda, I)$ and outputs 0 iff the experiment outputs 1. Similarly, if $b = 1$, then $\mathcal{B}$ perfectly simulates the experiment $\mathsf{Expt}^{\mathsf{CPA}}_{\widetilde{\mathsf{WE}},\mathcal{A}}(1^\lambda, I)$ and outputs 0 iff the experiment outputs 1. It then follows from Claim 8 and Eq. 3 that

$$\begin{aligned}
&\Pr[\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{B}}(1^\lambda, I) = 1]\\
&= \Pr[b = 0] \cdot \Pr[\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{B}}(1^\lambda, I) = 1 \mid b = 0]\\
&\quad + \Pr[b = 1] \cdot \Pr[\mathsf{Expt}^{\mathsf{KEM-CPA}}_{\mathsf{WKEM},\mathcal{B}}(1^\lambda, I) = 1 \mid b = 1]\\
&= \frac{1}{2} \cdot \left(\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda, I) = 1] + 1 - \Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\widetilde{\mathsf{WE}},\mathcal{A}}(1^\lambda, I) = 1]\right)\\
&\geq \frac{1}{2} \cdot \left(\frac{1}{2} + \epsilon(\lambda) + 1 - \frac{1}{2} - \mathsf{negl}(\lambda)\right)\\
&= \frac{1}{2} + \frac{1}{2}\epsilon(\lambda) - \mathsf{negl}(\lambda)
\end{aligned}$$ □

Finally, since WKEM is known to be extractable and $\frac{1}{2}\epsilon(\lambda) - \mathsf{negl}(\lambda)$ is a non-negligible function whenever $\epsilon(\lambda)$ is non-negligible, there exists a PPT extractor $\widetilde{\mathsf{Ext}}$, such that

$$\Pr\left[(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}} : \begin{array}{r} \mathtt{x} \leftarrow \mathcal{B}(1^\lambda, I) \\ \mathtt{w} \leftarrow \widetilde{\mathsf{Ext}}^{\mathcal{B}(\cdot)}(I, \mathtt{x}) \end{array}\right] \geq \delta(\lambda)$$

for some non-negligible function $\delta(\lambda)$. We can thus construct an extractor Ext for WE as follows. The extractor receives as input $(I, \mathtt{x}, m_0, m_1)$ and is given oracle access to $\mathcal{A}$. It then invokes $\widetilde{\mathsf{Ext}}^{\mathcal{B}}(\cdot)(I, \mathtt{x})$. Whenever $\widetilde{\mathsf{Ext}}$ queries $\mathtt{ct}_1$ to its oracle, Ext samples $b' \leftarrow \{0,1\}$, computes $\mathtt{ct}_2 \leftarrow \mathsf{Enc}^{\mathsf{sym}}(\mathtt{k}, m_{b'})$, and queries $(\mathtt{ct}_1, \mathtt{ct}_2)$ to its own oracle, forwarding the reply to $\widetilde{\mathsf{Ext}}$. It is easy to see that this perfectly simulates oracle access to $\mathcal{B}$. Therefore, it holds that

$$\Pr\left[(\mathtt{x}, \mathtt{w}) \in \mathcal{R}_{\mathcal{L}} : \begin{array}{r} (\mathtt{x}, m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, I) \\ \mathtt{w} \leftarrow \mathsf{Ext}^{\mathcal{A}(\cdot)}(I, \mathtt{x}, m_0, m_1) \end{array}\right] \geq \delta(\lambda)$$

as required. □

5 Laconic OT

As discussed in the introduction, we will now show how to construct a concretely efficient laconic OT protocol from our extractable witness encryption scheme for KZG commitments and openings.

Definition 20 ([10]). *A laconic oblivious transfer scheme is a tuple of PPT algorithms* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{H}, \mathsf{Send}, \mathsf{Receive})$ *defined as follows:*

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$**:** *The setup algorithm takes security parameter* λ *and database length* n *as input and returns public parameters* pp*.*

$(\mathtt{digest}, \mathtt{aux}) \leftarrow \mathsf{H}(\mathsf{pp}, D)$**:** *The hashing algorithm takes public parameters* pp *and database* D *as input and returns a public hash* $\mathtt{digest}$ *and some secret auxiliary information* $\mathtt{aux}$*.*

$c \leftarrow \mathsf{Send}(\mathsf{pp}, \mathtt{digest}, i, m_0, m_1)$**:** *The sender algorithm takes public parameters* pp*, database hash* $\mathtt{digest}$*, index* i*, and message* m_0 *and* m_1 *as input and returns message* c*.*

$m \leftarrow \mathsf{Receive}(\mathsf{pp}, \mathtt{aux}, c, i)$**:** *The receiver algorithm takes public parameters* pp*, auxiliary information* $\mathtt{aux}$*, sender message* m*, and index* i *as input and returns message* m*.*

Definition 21 (Correctness). *A laconic oblivious transfer scheme* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{H}, \mathsf{Send}, \mathsf{Receive})$ *is correct, if for all* $\lambda, n \in \mathbb{N}$ *with* $n = \mathsf{poly}(\lambda)$*, all* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$*, all databases* $D \in \{0,1\}^n$*, all* $(\mathtt{digest}, \mathtt{aux}) \leftarrow \mathsf{H}(\mathsf{pp}, D)$*, all* $i \in [n]$*, all* $m_0, m_1 \in \mathcal{M}$*, and all* $c \leftarrow \mathsf{Send}(\mathsf{pp}, \mathtt{digest}, i, m_0, m_1)$ *it holds that* $\mathsf{Receive}(\mathsf{pp}, \mathtt{aux}, c, i) = m_{D[i]}$*.*

Sender privacy requires the sender's message to the receiver to hide the message that was not selected by the receiver's choice bit.

Definition 22 (Sender Privacy). *A laconic oblivious transfer scheme* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{H}, \mathsf{Send}, \mathsf{Receive})$ *is sender private against semi-honest adversaries, if for all* $\lambda, n \in \mathbb{N}$ *with* $n = \mathsf{poly}(\lambda)$*, any PPT adversary* $\mathcal{A}$*, any database* $D \in \{0,1\}^n$*, any* $i \in [n]$*, and any pair of messages* $m_0, m_1 \in \mathcal{M}$*, it holds that*

$$\left| \begin{array}{l} \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{OT\text{-}S\text{-}Real}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1] \\ - \Pr[\mathsf{Expt}_{\mathcal{A}}^{\mathsf{OT\text{-}S\text{-}Sim}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1] \end{array} \right| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{OT\text{-}S\text{-}Real}}$ *and* $\mathsf{Expt}_{\mathcal{A}}^{\mathsf{OT\text{-}S\text{-}Sim}}$ *are defined as follows*

```
$\mathsf{Expt}_{\mathsf{LOT},\mathcal{A}}^{\mathsf{OT\text{-}S\text{-}Real}}(1^\lambda, 1^n, D, m_0, m_1, i)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
$(\mathtt{digest}, \mathtt{aux}) \leftarrow \mathsf{H}(\mathsf{pp}, D)$
$c \leftarrow \mathsf{Send}(\mathsf{pp}, \mathtt{digest}, i, m_0, m_1)$
$b \leftarrow \mathcal{A}(\mathsf{pp}, c, \mathsf{aux})$
return $b$
```

```
$\mathsf{Expt}_{\mathsf{LOT},\mathcal{A}}^{\mathsf{OT\text{-}S\text{-}Sim}}(1^\lambda, 1^n, D, m_0, m_1, i)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
$c \leftarrow \mathsf{Sim}(\mathsf{pp}, D, i, m_{D[i]})$
$b \leftarrow \mathcal{A}(\mathsf{pp}, c, \mathsf{aux})$
return $b$
```

In the original work of Cho et al. [10], no explicit definition for receiver privacy was stated. The authors informally argued that any protocol that is not receiver privacy can be transformed into one that is, via the use of generic secure computation. In our work, we do define receiver privacy and we focus on the arguably strongest possible notion, namely that of perfect receiver privacy, where the sender learns no information about the receivers choice bits in the information-theoretic sense. We do not make use of generic secure computation and instead our construction will directly satisfy this security notion.

Definition 23 (Receiver Privacy). *A laconic oblivious transfer scheme* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{H}, \mathsf{Send}, \mathsf{Receive})$ *is receiver private against semi-honest adversaries, if for all* $\lambda, n \in \mathbb{N}$ *with* $n = \mathsf{poly}(\lambda)$*, any PPT adversary* $\mathcal{A}$*, all databases* $D \in \{0,1\}^n$*, it holds that*

$$\left|\Pr[\mathsf{Expt}^{\mathsf{OT\text{-}R\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D) = 1] - \Pr[\mathsf{Expt}^{\mathsf{OT\text{-}R\text{-}Sim}}_{\mathcal{A}}(1^\lambda, 1^n, D) = 1]\right|,$$

where $\mathsf{Expt}^{\mathsf{OT\text{-}R\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}$ *and* $\mathsf{Expt}^{\mathsf{OT\text{-}R\text{-}Sim}}_{\mathsf{LOT},\mathcal{A}}$ *are defined as follows.*

```
$\mathsf{Expt}^{\mathsf{OT\text{-}R\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
$(\mathtt{digest}, \mathsf{aux}) \leftarrow \mathsf{H}(\mathsf{pp}, D)$
$b \leftarrow \mathcal{A}(\mathsf{pp}, \mathtt{digest})$
return $b$
```

```
$\mathsf{Expt}^{\mathsf{OT\text{-}R\text{-}Sim}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D)$
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
$\mathtt{digest} \leftarrow \mathsf{Sim}(\mathsf{pp})$
$b \leftarrow \mathcal{A}(\mathsf{pp}, \mathtt{digest})$
return $b$
```

The main property that makes laconic OT non-trivial and interesting is its efficiency. The digest is required to be independent of the original database size, the hashing should run in quasi-linear time in the database size, and both the computational complexity of sending and receiving should have at most a poly-logarithmic dependency on the size of the database.

Definition 24 (Efficiency). *A laconic oblivious transfer scheme* $\mathsf{LOT} = (\mathsf{Setup}, \mathsf{H}, \mathsf{Send}, \mathsf{Receive})$ *is efficient, if* $|\mathtt{digest}| \in \mathsf{poly}(\lambda)$ *and in particular independent of* $|D|$*, the hashing algorithm* H *runs in time* $|D| \cdot \mathsf{poly}(\log |D|, \lambda)$*, and both* Send *and* $\mathsf{Receive}$ *run in time* $\mathsf{poly}(\log |D|, \lambda)$*.*

5.1 Constructing Laconic OT

Our construction is conceptually very simple. The receiver will use a hiding vector commitment to compute a commitment $\mathtt{com}$ to their database D. Upon the i-th invocation of the OT, the sender will use our extractable witness encryption scheme to separately encrypt m_0 and m_1, such that they can be decrypted, if $\mathtt{com}$ can be opened to 0 and 1, respectively. The sender, who will receive the two ciphertexts, can then use their opening π_i for commitment $\mathtt{com}$ to decrypt the ciphertext containing $m_{D[i]}$. Sender privacy effectively follows from the security guarantees of the extractable witness encryption scheme, whereas receiver privacy follows from the hiding properties of the vector commitment.

```
$\mathsf{Setup}(1^\lambda, 1^n)$
    $\mathsf{pp} \leftarrow \mathsf{VC.Setup}(1^\lambda, 1^n)$

$\mathsf{H}(\mathsf{pp}, D)$
    $(\mathtt{com}, \overline{\mathtt{aux}}) \leftarrow \mathsf{VC.Commit}(\mathsf{pp}, D)$
    $(\pi_1, \ldots, \pi_n) \leftarrow \mathsf{BatchOpen}(\mathsf{pp}, \overline{\mathtt{aux}})$
    $\textbf{return } (\mathtt{com}, (D, \pi_1, \ldots, \pi_n))$

$\mathsf{Send}(\mathsf{pp}, \mathtt{digest}, i, m_0, m_1)$
    $\mathtt{ct}_0 \leftarrow \mathsf{WE.Enc}(\mathsf{pp}, (\mathtt{digest}, i, 0), m_0)$
    $\mathtt{ct}_1 \leftarrow \mathsf{WE.Enc}(\mathsf{pp}, (\mathtt{digest}, i, 1), m_1)$
    $\textbf{return } (\mathtt{ct}_0, \mathtt{ct}_1)$

$\mathsf{Receive}(\mathsf{pp}, \mathsf{aux}, (\mathtt{ct}_0, \mathtt{ct}_1), i)$
    $\textbf{parse } \mathsf{aux} \text{ as } (D, \pi_1, \ldots, \pi_n))$
    $b := D_i$
    $m_b \leftarrow \mathsf{WE.Dec}(\mathsf{pp}, \pi_i, \mathtt{ct}_b)$
    $\textbf{return } m_b$
```

Fig. 5. Laconic OT construction.

Theorem 10. *Let $\lambda, n \in \mathbb{N}$ with $n = \mathsf{poly}(\lambda)$. Let $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{BatchOpen}, \mathsf{Verify})$ be a correct, efficient, position binding, and perfectly weakly-hiding vector commitment. Let $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ be an extractable witness encryption scheme for $\mathcal{F}_{\mathsf{VC}}$. Then the construction in Fig. 5 is an efficient, sender-private, and perfectly receiver-private laconic OT.*

Proof. Efficiency follows immediately from the efficiency of VC. We will prove the other two properties separately.

Lemma 11. *Let $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ be a correct and position binding vector commitment. Let $\mathsf{WE} = (\mathsf{Enc}, \mathsf{Dec})$ be an extractable witness encryption scheme for $\mathcal{F}_{\mathsf{VC}}$. Then the construction in Fig. 5 is sender private.*

Proof. We specify a simulator as follows. On input $\mathsf{pp}, D, i, m_{D[i]}$ the simulator simply computes

$$\begin{aligned} &\mathtt{ct}_{D[i]} \leftarrow \mathsf{WE.Enc}(\mathsf{pp}, (\mathtt{digest}, i, D[i]), m_{D[i]}) \\ \text{and}\quad &\mathtt{ct}_{1-D[i]} \leftarrow \mathsf{WE.Enc}(\mathsf{pp}, (\mathtt{digest}, i, 1 - D[i]), 0) \end{aligned}$$

and outputs $(\mathtt{ct}_0, \mathtt{ct}_1)$.

Let $D \in \{0,1\}^n$ be an arbitrary database, $i \in [n]$ an arbitrary index, $m_0, m_1 \in \mathbb{F}$ an arbitrary pair of messages, and $\mathcal{A}$ an arbitrary PPT adversary against sender privacy with

$$\begin{aligned} \epsilon(\lambda) = &\Pr[\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1] \\ &- \Pr[\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Sim}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1]. \end{aligned} \tag{4}$$

Assume wlog, that $\epsilon(\lambda) \geq 0$.

We construct a PPT adversary $\mathcal{B}$ against the extractability of WE as follows. On input $1^\lambda, \mathsf{ck}$, the adversary $\mathcal{B}$ computes $(\mathtt{com}, \overline{\mathtt{aux}}) \leftarrow \mathsf{VC.Commit}(\mathsf{pp}, D)$ and outputs

$$\big(\underbrace{(\texttt{com}, i, 1 - D[i])}_{:=\text{x}}, 0, m_{1-D[i]}\big).$$

Upon receiving $\texttt{ct}_{1-D[i]}$, $\mathcal{B}$ further computes

$$\texttt{ct}_{D[i]} \leftarrow \mathsf{WE.Enc}(\mathsf{ck}, (\texttt{digest}, i, D[i]), m_{D[i]}),$$

sets $\texttt{aux} := (D, \texttt{com}, \overline{\texttt{aux}})$ and invokes $\mathcal{A}(\mathsf{ck}, (\texttt{ct}_0, \texttt{ct}_1), \texttt{aux})$. Eventually $\mathcal{A}$ will output a bit, which $\mathcal{B}$ will also output.

Let $b \in \{0,1\}$ be the bit in the experiment $\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{B}}(1^\lambda)$. It is easy to see that, if $b = 1$, then $\mathcal{B}$ perfectly simulates $\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i)$. On the other hand, if $b = 0$, then $\mathcal{B}$ perfectly simulates $\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Sim}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i)$. It thus follows that

$$\begin{aligned}
&\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{B}}(1^\lambda, \mathsf{ck}) = 1]\\
&= \frac{1}{2}\big(\Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda) = 1 \mid b = 1] + \Pr[\mathsf{Expt}^{\mathsf{CPA}}_{\mathsf{WE},\mathcal{A}}(1^\lambda) = 1 \mid b = 0]\big)\\
&= \frac{1}{2}\left(\begin{array}{r}\Pr[\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1]\\ + \Pr[\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Sim}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i) = 0]\end{array}\right)\\
&= \frac{1}{2}\left(\begin{array}{r}\Pr[\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Real}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1]\\ +1 - \Pr[\mathsf{Expt}^{\mathsf{OT\text{-}S\text{-}Sim}}_{\mathsf{LOT},\mathcal{A}}(1^\lambda, 1^n, D, m_0, m_1, i) = 1]\end{array}\right) = \frac{1}{2} + \frac{1}{2}\epsilon(\lambda).
\end{aligned}$$

Since WE is extractable, it follows that there exists a PPT algorithm Ext, such that, if $\epsilon(\lambda)$, and thereby also $\epsilon(\lambda)/2$ were non-negligible, it would hold that

$$\Pr\left[\mathsf{Verify}(\mathsf{ck}, \texttt{com}, \pi, i, 1 - D[i]) = 1 : \begin{array}{r}((\texttt{com}, i, 1 - D[i]), 0, m_{1-D[i]}) \leftarrow \mathcal{B}(1^\lambda, \mathsf{ck})\\ \pi \leftarrow \mathsf{Ext}^{\mathcal{B}(\cdot)}(\mathsf{ck}, (\texttt{com}, i, 1 - D[i]), 0, m_{1-D[i]})\end{array}\right]$$
$$\geq \delta(\lambda)$$

for some non-negligible function $\delta(\lambda)$.

To show that this can't be the case, we construct an adversary $\mathcal{C}$ against the position binding of VC as follows. On input ck, the adversary $\mathcal{C}$ computes $(\texttt{com}, \overline{\texttt{aux}}) \leftarrow \mathsf{VC.Commit}(\mathsf{pp}, D)$, and invokes $\mathsf{Ext}^{\mathcal{B}(\cdot)}((\texttt{com}, i, 1 - D[i]), 0, m_{1-D[i]})$. Whenever Ext queries its oracle with $\texttt{ct}_{1-D[i]}$, $\mathcal{C}$ further computes $\texttt{ct}_{D[i]} \leftarrow \mathsf{WE.Enc}(\mathsf{ck}, (\texttt{digest}, i, D[i]), m_{D[i]})$, sets $\texttt{aux} := (D, \texttt{com}, \overline{\texttt{aux}})$, invokes $b \leftarrow \mathcal{A}(\mathsf{ck}, (\texttt{ct}_0, \texttt{ct}_1), \texttt{aux})$ and replies with b. Eventually, Ext will output π. $\mathcal{C}$ will then compute $\pi' \leftarrow \mathsf{VC.Open}(\mathsf{ck}, \overline{\texttt{aux}}, i)$ and output $\texttt{com}, i, 1 - D[i], D[i], \pi, \pi'$.

It is easy to see, that $\mathcal{C}$ perfectly simulates $\mathcal{B}$ for the extractor. Therefore, it holds with probability at least $\delta(\lambda)$, that $\mathsf{Verify}(\mathsf{ck}, \texttt{com}, \pi, i, 1 - D[i]) = 1$. The correctness of the vector commitment guarantees that $\mathsf{Verify}(\mathsf{ck}, \texttt{com}, \pi', i, D[i]) = 1$. Since further $D[i] \neq 1 - D[i]$, it thus holds that

$$\mathsf{negl}(\lambda) \geq \Pr\left[\begin{array}{c} m_0 \neq m_1 \\ \wedge \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi_0, i, m_0) = 1 \\ \wedge \mathsf{Verify}(\mathsf{ck}, \mathtt{com}, \pi_1, i, m_1) = 1 \end{array} : \begin{array}{r} \mathsf{ck} \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (\mathtt{com}, i, m_0, m_1, \pi_0, \pi_1) \leftarrow \mathcal{C}(\mathsf{ck}) \end{array}\right]$$
$$= \delta(\lambda)$$

Which immediately implies that $\epsilon(\lambda) \leq \mathsf{negl}(\lambda)$ as required. □

Lemma 12. *Let* $\mathsf{VC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$ *be a perfectly weakly hiding vector commitment. Then the construction in Fig. 5 is perfectly receiver private.*

Proof. The simulator takes as input the public parameters $\mathtt{pp} = \mathsf{ck}$, computes $(\mathtt{com}, \overline{\mathtt{aux}}) \leftarrow \mathsf{VC.Commit}(\mathsf{ck}, 0^n)$ and outputs $\mathtt{com}$. The only difference between the two experiments from the point of view of an adversary is that in one case they receive a commitment to D, whereas in the other they receive a commitment to 0^n. However, since VC is perfectly weakly hiding, those two commitments are distributed identically. □

Now Theorem 10 follows directly by combining Lemmas 11 and 12. □

On Actively Secure Laconic OT. While we do not explore this question in our current work, we would still like to highlight that our laconic OT construction provides some active security guarantees, at least on an intuitive level. No matter what the sender chooses as their polynomial commitment, at least one of the sender's messages in each invocation will always be indistinguishable from random for the receiver. This follows from the fact that the polynomial commitment scheme is binding. Whether it can be made to satisfy a simulation-based active security notion is left as an open problem for future work.

6 Benchmarks

The laconic OT protocol from Sect. 5 was instantiated with the KZG based vector commitment from Sect. 2.5 and the extractable witness commitment derived by combining the extractable witness KEM from Sect. 3 with the generic transformation from Sect. 4. A simple one-time pad as the symmetric encryption scheme in the transformation. The full construction was implemented[4] in Rust using `arkworks` [12]. The `BLS12-381` [6] curve was used as the pairing-friendly elliptic curve throughout the implementation. All involved computational costs and bandwidth overheads were measured for various parameters. All benchmarks were run on a personal laptop with an `i7-11800H @ 2.30GHz` CPU and 64 GB of RAM.

[4] https://github.com/rot256/research-we-kzg.

As already explained in Sect. 2.5, the batch opening technique of Feist and Khovratovich was used [14] to precompute all openings of the used vector commitment during the hashing of the database of choice indices. For concreteness the sender's OT inputs are assumed to be 256-bit messages. A single witness encryption for such a message consists of a 96 byte witness KEM ciphertext and a 32 bytes large one-time pad encryption of the message itself, thus leading to a 128 byte ciphertext. The public parameters for a database of size n consists of $n+1$ many $\mathbb{G}_1$ elements and two $\mathbb{G}_2$ elements with $\mathbb{G}_1$ and $\mathbb{G}_2$ for BLS12-381 clocking in at 48 and 96 bytes respectively. The benchmark results can be found in Table 1.

Table 1. Runtimes and bandwidth overheads of our laconic OT protocol for varying sizes of the receiver's database D. The computation times for $|D| = 2^{31}$ are extrapolated to compare with [23].

$\|D\|$	Sizes			Times		
	`pp`	`digest`	Sender Msg.	Hash	Send	Receive
2^6	3.2 KB	48 B	256 B	173 ms	4 ms	1 ms
2^8	12.2 KB	48 B	256 B	723 ms	4 ms	1 ms
2^{10}	48.2 KB	48 B	256 B	3 s	4 ms	1 ms
2^{12}	192.2 KB	48 B	256 B	10 s	4 ms	1 ms
2^{14}	768.2 KB	48 B	256 B	43 s	4 ms	1 ms
2^{16}	3.0 MB	48 B	256 B	3 min	5 ms	1 ms
2^{18}	12.0 MB	48 B	256 B	8 min	5 ms	1 ms
2^{31}	96.0 GB	48 B	256 B	—	5 ms	1 ms

Our construction is highly efficient in most parameters, in particular database digest and the sender's message are both not just constant in size but concretely very small. The time to compute the sender's message and to decode on the receiver's side in an OT invocation is below 5 ms. The main drawbacks of our construction the initial one-time cost of computing the hash of the receiver's database and the size of the public parameters.

Weakly Laconic OT. We also examine the efficiency of a variant of our construction, which does not strictly satisfy the formal efficiency requirements for laconic OT protocols from Definition 24, but has significantly smaller public parameters at the cost of somewhat larger digests. The idea for achieving this trade-off is to simply partition the receiver's database D into $\sqrt{|D|}$ smaller databases and to then hash each of them individually. The hash returned by the receiver is simply the concatenation of all $\sqrt{|D|}$ hashes. This construction does not formally satisfy the asymptotic efficiency requirement of a laconic OT because the digest size now (sublinearly) depends on the size of the input database. The main observation here is that all of those hashes can be computed using the same trusted setup for instances of size $\sqrt{|D|}$. This simple trick was already observed by Green, Jain, and Van Laer in their work [23]. We calculate the sizes of objects and we extrapolate the timing benchmarks for this construction from our benchmarks for regular laconic OT. The results can be found in Table 2.

Table 2. Runtimes and bandwidth overheads for the variant of our laconic OT protocol, which splits the database D into $\sqrt{|D|}$ many smaller databases. The numbers are extrapolated from our experimental results for laconic OT.

$\|D\|$	Sizes			Times		
	`pp`	`digest`	Sender Msg.	Hash	Send	Receive
2^6	624 B	384 B	256 B	194 ms	4 ms	1 ms
2^8	1008 B	768 B	256 B	743 ms	4 ms	1 ms
2^{10}	1.7 KB	1.5 KB	256 B	3 s	4 ms	1 ms
2^{12}	3.2 KB	3.0 KB	256 B	11 s	4 ms	1 ms
2^{14}	6.2 KB	6.0 KB	256 B	44 s	4 ms	1 ms
2^{16}	12.2 KB	12.0 KB	256 B	3 min	4 ms	1 ms
2^{18}	24.2 KB	24.0 KB	256 B	12 min	4 ms	1 ms
2^{31}	3 MB	1.5 MB	256 B	69 days	4 ms	1 ms

Comparison to Related Works. Equipped with the concrete performance numbers for our constructions, we can compare its efficiency to that of existing protocols. Green, Jain, Van Laer [23] provide benchmarks for a single data point for their laconic OT construction, which does not achieve sender privacy. They argue that one can generically obtain sender privacy by using garbled circuits, which would incur a significant overhead on top of their provided benchmarks. In their favour, we shall ignore this and simply compare it to our construction that achieves both sender and receiver privacy.

For a database size of 2^{31}, their construction's public parameters 412.3 GB, their digest is 48 B, their sender's message size is 1.34 KB and receiving takes 27.7 *minutes*. That means their public parameters are $\approx$ 4x larger, their digests have the same size, their sender's message is $\approx$ 5x larger, and their receiving time is larger by 6 *orders of magnitude*. No further data points were provided in their benchmarks.

The authors also provide benchmark results for a weakly laconic version of their construction, which splits the database in $\sqrt{|D|}$ chunks as described above. Comparing it to our weakly laconic OT construction, they have $\approx$ 3x larger public parameters, comparable digest size, and $\approx$ 5x larger sender's message. Their receiver's computational time is still $\approx$ 38x larger.

Döttling et al. [13] only sketch how to construct laconic OT from another cryptographic primitive, which they do provide some benchmarks for. The main advantage of their work is that the public parameters do not grow with the receiver's database size. Taking the benchmarks for their lowest security levels, their public parameters are 8 MB, their digest is 2 KB, and their sender's message size is 98 MB. The constant size of their public parameters makes them concretely smaller than ours once the databases size reaches 2^{18}. Their digests are always $\approx$ 8x larger and their sender's message size is 5 *orders of magnitude* larger.

Comparing with our weakly laconic OT protocol, our public parameters would only become concretely larger than theirs, when the database size reaches at least 2^{35}. For a database size of 2^{18}, our digest is $\approx$ 12x larger, but our sender's message is still 5 orders of magnitude smaller. We believe that a slightly larger digest size in exchange for much smaller sender messages is a valuable trade-off. The digest is sent once, but the sender's messages need to be send for each OT invocation.

On the Importance of Small Sender Messages. The benchmarks consider several different parameters, each of which has its own significance when it comes to the overall efficiency of a laconic OT protocol. It is, however, worth highlighting the importance of one particular parameter, namely the sender's message size. Laconic OT is most useful when many OT invocations are required. The digest is sent *once*, whereas the sender's message is sent *once per invocation*. Thus, it makes sense to not only look at the size of a single sender message, but at the total communication cost over many OT invocations. For a database size of 2^{18}, the total communication cost is $\approx$ 343 MB in the work of Green, Jain, Van Laer [23] and $\approx$ 24.5 terabyte in the work of Döttling et al. [13]. In our construction, the total communication cost is $\approx$ 64 MB.

Acknowledgements. We would like to thank Nathan Xiong for pointing out a flaw in an earlier version of the proof of Claim 5.

References

1. Aranha, D.F., Lin, C., Orlandi, C., Simkin, M.: Laconic private set-intersection from pairings. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 111–124. ACM Press (Nov 2022). https://doi.org/10.1145/3548606.3560642
2. 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.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Berlin, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8_1
3. Benhamouda, F., Jain, A., Komargodski, I., Lin, H.: Multiparty reusable non-interactive secure computation from LWE. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part II. LNCS, vol. 12697, pp. 724–753. Springer, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-77886-6_25

4. Benhamouda, F., Lin, H.: k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part II. LNCS, vol. 10821, pp. 500–532. Springer, Cham (Apr / May 2018). https://doi.org/10.1007/978-3-319-78375-8_17
5. Benhamouda, F., Lin, H.: Mr NISC: Multiparty reusable non-interactive secure computation. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 349–378. Springer, Cham (Nov 2020). https://doi.org/10.1007/978-3-030-64378-2_13
6. Bowe, S.: Bls12-381: New zk-snark elliptic curve construction (Mar 2017), https://electriccoin.co/blog/new-snark-curve/
7. Campanelli, M., Fiore, D., Khoshakhlagh, H.: Witness encryption for succinct functional commitments and applications. Cryptology ePrint Archive, Report 2022/1510 (2022), https://eprint.iacr.org/2022/1510
8. Catalano, D., Fiore, D.: Vector commitments and their applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 55–72. Springer, Berlin, Heidelberg (Feb / Mar 2013). https://doi.org/10.1007/978-3-642-36362-7_5
9. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., Ward, N.P.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 738–768. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45721-1_26
10. Cho, C., Döttling, N., Garg, S., Gupta, D., Miao, P., Polychroniadou, A.: Laconic oblivious transfer and its applications. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part II. LNCS, vol. 10402, pp. 33–65. Springer, Cham (Aug 2017). https://doi.org/10.1007/978-3-319-63715-0_2
11. Choi, G., Vaudenay, S.: Towards witness encryption without multilinear maps - extractable witness encryption for multi-subset sum instances with no small solution to the homogeneous problem. In: Park, J.H., Seo, S.H. (eds.) ICISC 21. LNCS, vol. 13218, pp. 28–47. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-031-08896-4_2
12. arkworks contributors: `arkworks` zksnark ecosystem (2022), https://arkworks.rs
13. Döttling, N., Kolonelos, D., Lai, R.W.F., Lin, C., Malavolta, G., Rahimi, A.: Efficient laconic cryptography from learning with errors. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 417–446. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30620-4_14
14. Feist, D., Khovratovich, D.: Fast amortized KZG proofs. Cryptology ePrint Archive, Report 2023/033 (2023), https://eprint.iacr.org/2023/033
15. 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, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_2
16. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: Permutations over Lagrange-bases for oecumenical noninteractive arguments of knowledge. Cryptology ePrint Archive, Report 2019/953 (2019), https://eprint.iacr.org/2019/953

17. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS. pp. 40–49. IEEE Computer Society Press (Oct 2013). https://doi.org/10.1109/FOCS.2013.13
18. Garg, S., Gentry, C., Halevi, S., Wichs, D.: On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 518–535. Springer, Berlin, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2_29
19. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 467–476. ACM Press (Jun 2013). https://doi.org/10.1145/2488608.2488667
20. Garg, S., Srinivasan, A.: Garbled protocols and two-round MPC from bilinear maps. In: Umans, C. (ed.) 58th FOCS. pp. 588–599. IEEE Computer Society Press (Oct 2017). https://doi.org/10.1109/FOCS.2017.60
21. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run Turing machines on encrypted data. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 536–553. Springer, Berlin, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_30
22. Gorbunov, S., Reyzin, L., Wee, H., Zhang, Z.: Pointproofs: Aggregating proofs for multiple vector commitments. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 2007–2023. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3417244
23. Green, M., Jain, A., Laer, G.V.: Efficient set membership encryption and applications. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 1080–1092. ACM Press (Nov 2023). https://doi.org/10.1145/3576915.3623131
24. 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, Berlin, Heidelberg (Dec 2010). https://doi.org/10.1007/978-3-642-17373-8_11
25. Libert, B., Passelègue, A., Riahinia, M.: PointProofs, revisited. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part IV. LNCS, vol. 13794, pp. 220–246. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22972-5_8
26. Maller, M., Bowe, S., Kohlweiss, M., Meiklejohn, S.: Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2111–2128. ACM Press (Nov 2019). https://doi.org/10.1145/3319535.3339817
27. Rabin, M.O.: How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, (1981), http://eprint.iacr.org/2005/187
28. Srinivasan, S., Chepurnoy, A., Papamanthou, C., Tomescu, A., Zhang, Y.: Hyperproofs: Aggregating and maintaining proofs in vector commitments. In: Butler, K.R.B., Thomas, K. (eds.) USENIX Security 2022. pp. 3001–3018. USENIX Association (Aug 2022)
29. Tomescu, A., Abraham, I., Buterin, V., Drake, J., Feist, D., Khovratovich, D.: Aggregatable subvector commitments for stateless cryptocurrencies. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 45–64. Springer, Cham (Sep 2020). https://doi.org/10.1007/978-3-030-57990-6_3

30. Tsabary, R.: Candidate witness encryption from lattice techniques. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, vol. 13507, pp. 535–559. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15802-5_19
31. Vaikuntanathan, V., Wee, H., Wichs, D.: Witness encryption and null-IO from evasive LWE. In: Agrawal, S., Lin, D. (eds.) ASIACRYPT 2022, Part I. LNCS, vol. 13791, pp. 195–221. Springer, Cham (Dec 2022). https://doi.org/10.1007/978-3-031-22963-3_7
32. Wang, W., Ulichney, A., Papamanthou, C.: BalanceProofs: Maintainable vector commitments with fast aggregation. In: Calandrino, J.A., Troncoso, C. (eds.) USENIX Security 2023. pp. 4409–4426. USENIX Association (Aug 2023)

Revisiting Pairing-Friendly Curves with Embedding Degrees 10 and 14

Yu Dai[1], Debiao He[2], Cong Peng[2](✉), Zhijian Yang[1,3], and Chang-an Zhao[4,5](✉)

[1] School of Mathematics and Statistics, Wuhan University, Wuhan, China
eccdaiy39@gmail.com, zjyang.math@whu.edu.cn

[2] School of Cyber Science and Engineering, Wuhan University, Wuhan, China
{hedebiao,cpeng}@whu.edu.cn

[3] Institute for Math & AI, Wuhan University, Wuhan, China

[4] School of Mathematics, Sun Yat-sen University, Guangzhou, China
zhaochan3@mail.sysu.edu.cn

[5] Guangdong Key Laboratory of Information Security, Guangzhou, China

Abstract. Since 2015, there has been a significant decrease in the asymptotic complexity of computing discrete logarithms in finite fields. As a result, the key sizes of many mainstream pairing-friendly curves have to be updated to maintain the desired security level. In PKC'20, Guillevic conducted a comprehensive assessment of the security of a series of pairing-friendly curves with embedding degrees ranging from 9 to 17. In this paper, we focus on five pairing-friendly curves with embedding degrees 10 and 14 at the 128-bit security level, with BW14-351 emerging as the most competitive candidate. First, we extend the optimized formula for the optimal pairing on BW13-310, a 128-bit secure curve with a prime p in 310 bits and embedding degree 13, to our target curves. This generalization allows us to compute the optimal pairing in approximately $\log r/(2\varphi(k))$ Miller iterations, where r and k are the order of pairing groups and the embedding degree respectively. Second, we develop optimized algorithms for cofactor multiplication for $\mathbb{G}_1$ and $\mathbb{G}_2$, as well as subgroup membership testing for $\mathbb{G}_2$ on these curves. Finally, we provide detailed performance comparisons between BW14-351 and other popular curves on a 64-bit platform in terms of pairing computation, hashing to $\mathbb{G}_1$ and $\mathbb{G}_2$, group exponentiations, and subgroup membership testings. Our results demonstrate that BW14-351 is a strong candidate for building pairing-based cryptographic protocols.

Keywords: pairing-friendly curves · BW14-351 · the 128-bit security level

1 Introduction

The past two decades have witnessed the application of elliptic curve pairings in public-key cryptosystems, such as Direct Anonymous Attestation (DAA) [13, 51], Succinct Non-interactive Arguments of Knowledge (SNARKs) [3,21,22,30], and Verifiable Delay Function(VDF) [20]. A cryptographic pairing is a non-degenerate bilinear map defined as $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where the three pairing groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ have the same large prime order r. Specifically, $\mathbb{G}_1$ and $\mathbb{G}_2$ are two independent subgroups of an elliptic curve E over a finite field $\mathbb{F}_{p^k}$, while $\mathbb{G}_T$ is a subgroup of the multiplicative group $\mathbb{F}_{p^k}^*$. The value of k is the smallest positive integer such that $E[r] \subseteq E(\mathbb{F}_{p^k})$.

The security of pairing-based cryptographic protocols relies on the hardness of the discrete logarithm problem (DLP) in the three pairing groups. The best-known attack algorithm for solving the DLP on an elliptic curve (ECDLP) in the two input pairing groups $\mathbb{G}_1$ and $\mathbb{G}_2$ is the Pollard rho algorithm [43], which requires around $\sqrt{r}$ group operations. Thus, the size of the prime r is at least 256 bits for reaching the 128-bit security level. As for the DLP on a finite field (FFDLP) $\mathbb{F}_{p^k}$ in $\mathbb{G}_T$ whose characteristic p is not small, the best-known algorithm is the number field sieve (NFS) [44]. According to the standardization reported by ENISA [1] in 2013, a 3072-bit finite field is 128-bit secure. Since then, a series of variants of NFS have been proposed [9,36,38], resulting in a drastic decrease for the security level of mainstream pairing-friendly curves. In particular, Kim and Barbulescu [38] proposed the special extended tower number field sieve (SexTNFS), which is applied to a composite extension field whose characteristic p can be parameterized by a tiny-coefficients polynomial of moderate degree. This variant is almost tailored to mainstream pairing-friendly curves, such as the Barreto-Naehrig(BN) [11] and Barreto-Lynn-Scott(BLS) [11] families. For example, the estimates in [8,32] suggest that the updated security level of the previous 128-bit secure BN curve has dropped down to $100 \sim 103$ bits.

In PKC'20, Guillevic [31] analyzed the consequence of the improvement of NFS in detail and recommended a list of pairing-friendly curves with embedding degrees from 10 to 16. In particular, Guillevic pointed out that the size of the prime p on both BN and BLS12 curves has to be increased to 446 bits to match the updated 128-bit security level, and the BLS12-446 curve is the most efficient choice for pairing computation at this security level across different pairing-friendly curves. However, owing to the large size of the characteristic p, both BLS12-446 and BN446 suffer a performance penalty concerning operations associated with $\mathbb{G}_1$. Therefore, two new curves derived from [24, Construction 6.6] have emerged for fast group exponentiation in $\mathbb{G}_1$: BW13-310 and BW19-286 [15]. Recently, Dai, Zhang and Zhao [18] proposed a new formula for computing pairing on BW13-310. More specifically, the number of iterations in Miller's algorithm on the curve is only around $\log r/(2\varphi(k))$. However, due to the lack of twists, the trick of denominator elimination is no longer applicable. In other words, even though the length of the Miller loop on BW13-310 is extremely short, the computational cost for each Miller doubling/addition step is expensive. In addition, since the group $\mathbb{G}_2$ on BW13-310 is defined over the full extension field $\mathbb{F}_{p^{13}}$, the operations associated with $\mathbb{G}_2$ are costly, such as hashing to $\mathbb{G}_2$ and group exponentiation in $\mathbb{G}_2$. It motivates us to search for new pairing-friendly curves such that the Miller loop can be performed in $\log r/(2\varphi(k))$ iterations, and the trick of denominator elimination applies as well.

1.1 Our Contributions

In this work, we revisit the cyclotomic pairing-friendly curves presented in [24] with embedding degrees 10 and 14. A comprehensive research is presented that aims to facilitate the implementation of pairing-based cryptographic protocols using these curves. Our contributions are summarized as follows.

- We generalize the optimized formula for the optimal pairing on BW13-310 to our target curves. Specifically, the automorphism action can be extracted from the Miller function evaluation, reducing the number of Miller iterations to approximately $\log r/(2\varphi(k))$. In addition, we refine the best-known algorithm for the final exponentiation to save several field multiplications.
- We develop new algorithms for key building blocks involved in implementing pairing-based protocols on our target curves, including cofactor multiplication for $\mathbb{G}_1$ and $\mathbb{G}_2$, and subgroup membership testing for $\mathbb{G}_2$.
- Utilizing the RELIC toolkit [2], we present high-speed software implementations of pairing computation, hashing to $\mathbb{G}_1$ and $\mathbb{G}_2$, group exponentiations, and subgroup membership testings over two target curves named BW10-511 and BW14-351 on a 64-bit platform. Our results show that compared to popular curves at the updated 128-bit security level, including BLS12-446, BN446, and BW13-310, BW14-351 is competitive for building pairing-based cryptographic protocols. In more detail,
  - the performance of pairing computation on BW14-351 is even slightly faster than that on BN446 and BW13-310, while about 16.2% slower than that on BLS12-446;
  - in terms of group exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_T$, BW14-351 is about 49.4% and 20.4% faster than BLS12-446, 118.5% and 100% faster than BN446, while 35.1% and 3.4% slower than BW13-310;
  - compared to BW13-310, BW14-351 incurs a lighter penalty for hashing to $\mathbb{G}_2$ and group exponentiation in $\mathbb{G}_2$, while is still slower than BN446 and BLS12-446.

Code: Our code is available at https://github.com/eccdaiy39/BW10-14.

2 Preliminaries

In this section, we recall some basic properties of ordinary elliptic curves, pairings and endomorphisms.

2.1 Ordinary Elliptic Curves over Finite Fields

Let $\mathbb{F}_p$ be a prime field with characteristic $p > 3$. Let E be an elliptic curve over $\mathbb{F}_p$ of the form $y^2 = x^3 + ax + b$, where $a, b \in \mathbb{F}_p$ such that $4a^3 + 27b^2 \neq 0$. The j-invariant of E is defined as $j(E) = 1728\frac{4a^3}{4a^3+27b^2}$. We denote by $E(\mathbb{F}_p)$ the group of $\mathbb{F}_p$-rational points of E, together with the identity element $\mathcal{O}_E$. Then the order of $E(\mathbb{F}_p)$ is given by $\#E(\mathbb{F}_p) = p + 1 - t$, where t is the trace of the

Frobenius endomorphism $\pi : (x, y) \rightarrow (x^p, y^p)$. If $t \neq 0$, then the curve E is said to be *ordinary*, and *supersingular* otherwise. Let r be a large prime divisor of $\#E(\mathbb{F}_p)$, and let $E[r]$ denote the r-torsion subgroup of E. The embedding degree k of E with respect to r and p is the smallest positive integer such that $E[r] \subseteq E(\mathbb{F}_{p^k})$. If $k > 1$, then k is the smallest integer such that $r \mid p^k - 1$.

An endomorphism α of E over $\bar{\mathbb{F}}_p$ is a non-constant rational map from E to itself over $\bar{\mathbb{F}}_p$, where $\bar{\mathbb{F}}_p$ is the algebraic closure of $\mathbb{F}_p$. The set of all endomorphisms of E over $\bar{\mathbb{F}}_p$, together with the zero map defined by $0(P) = \mathcal{O}_E$, forms a ring denoted as $\mathrm{End}(E)$. We denote by K the imaginary quadratic field $K = \mathbb{Q}(\sqrt{-D})$, where D is the square-free part of $4p - t^2$. Let O_K be the largest subring of K. Since E is ordinary, $\mathrm{End}(E)$ is an order in O_K, i.e., $\mathbb{Z}[\pi] \subseteq \mathrm{End}(E) \subseteq O_K$. For any $\alpha \in \mathrm{End}(E)$, the characteristic equation of α can be represented as $x^2 + mx + n = 0$ for two integers m and n, where n is called the norm of α, i.e., $\mathrm{Nrd}(\alpha) = n$. In particular, the characteristic equation of π is given as $\pi^2 - t\pi + p = 0$. For each endomorphism α, there is a unique endomorphism $\hat{\alpha}$ such that $\alpha \circ \hat{\alpha} = \mathrm{Nrd}(\alpha)$, which is called the dual of α.

Let $\mathrm{Aut}(E)$ be the automorphism group of E, and let $d = \gcd(k, \#\mathrm{Aut}(E))$. If $d > 1$, then there exists a unique degree-d twist E' such that $r \mid \#E'(\mathbb{F}_{p^{k/d}})$ with an untwisting isomorphism ϕ: $E' \rightarrow E$. In elliptic curve cryptography, ordinary elliptic curves with j-invariant 0 or 1728 are particularly interesting as they are equipped with an efficiently computable endomorphism. More precisely,

- if $j(E) = 0$, then we have $a = 0$ and $p \equiv 1 \bmod 3$ [50, Proposition 4.33]. There exists an endomorphism $E \rightarrow E$ given as $\tau : (x, y) \rightarrow (\omega \cdot x, y)$, where ω is a primitive cube root of unity in $\mathbb{F}_p^*$. The characteristic equation of τ is $\tau^2 + \tau + 1 = 0$ and the dual of τ is $\hat{\tau} : (x, y) \rightarrow (\omega^2 \cdot x, y)$;
- if $j(E) = 1728$, then we have $b = 0$ and $p \equiv 1 \bmod 4$ [50, Theorem 4.23]. There exists an endomorphism $E \rightarrow E$ given as $\tau : (x, y) \rightarrow (-x, i \cdot y)$, where i is a primitive fourth root of unity in $\mathbb{F}_p^*$. The characteristic equation of τ is $\tau^2 + 1 = 0$ and the dual of τ is $\hat{\tau} : (x, y) \rightarrow (-x, -i \cdot y)$.

For the two types of curves, there exist the following two endomorphisms on E':

$$\eta = \phi^{-1} \circ \tau \circ \phi, \psi = \phi^{-1} \circ \pi \circ \phi,$$

where η and ψ have the same characteristic equations as τ and π, respectively.

2.2 Optimal Pairing

Given a random point $Q \in E(\mathbb{F}_{p^k})$ and an integer m, a Miller function $f_{m,Q}$ is a normalized rational function in $\mathbb{F}_{p^k}(E)$ with divisor

$$div(f_{m,Q}) = m(Q) - ([m]Q) - (m - 1)(\mathcal{O}_E). \tag{1}$$

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be respectively 1- and p-eigenspaces of π acting on $E[r]$, i.e., $\mathbb{G}_1 = E(\mathbb{F}_p)[r]$ and $\mathbb{G}_2 = E[r] \cap \mathrm{Ker}(\pi - [p])$. Let $\mathbb{G}_T$ be the group of r-th roots of unity in $\mathbb{F}_{p^k}^*$. Let $\lambda = \sum_{i=0}^{L} c_i p^i$ be a multiple of the prime r with $c_i \in \mathbb{Z}$ for each i. Then, the general expression of the optimal pairing [49, Theorem 7] on E is given as:

$$e : \mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{G}_T,$$
$$(Q,P) \to \left(\prod_{i=0}^{L} f_{c_i,Q}^{p^i}(P) \cdot \prod_{i=0}^{L-1} \frac{\ell_{[s_{i+1}]Q,[c_i p^i]Q}(P)}{\nu_{[s_i]Q}(P)} \right)^{\frac{(p^k-1)}{r}}, \tag{2}$$

where $s_i = \sum_{j=i}^{L} c_j p^j$, $\ell_{[i]R,[j]R}$ is the straight line passing through $[i]R$ and $[j]R$, and $\nu_{[i+j]R}$ is the vertical line passing through $[i+j]R$. The Miller function $f_{c_i,Q}$ evaluated at the point P for each i can be obtained by executing Miller's algorithm [42], which is described in **Alg.** 1. Vercauteren [49, Theorem 7] proved that there exists a short vector $(c_0, c_1, \cdots, c_L)$ satisfying that $\max |c_i| \approx r^{1/\varphi(k)}$. Thus, the optimal pairing can be computed in approximately $\log r/\varphi(k)$ Miller iterations. Moreover, if the embedding degree k is even, the vertical line evaluations can be ignored because these values lie in the subfield $\mathbb{F}_{p^{k/2}}$ and can be killed by the exponentiation by $(p^k - 1)/r$.

Algorithm 1: Miller's Algorithm

```
Input: $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, $m = \sum_{i=0}^{L} m_i 2^i$ with $m_i \in \{-1, 0, 1\}$
Output: $f_{m,Q}(P)$
1: $T \leftarrow Q$, $f \leftarrow 1$
2: for $i = L - 1$ down to 0 do
3:     $f \leftarrow f^2 \cdot \frac{\ell_{T,T}(P)}{\nu_{[2]T}(P)}$, $T \longleftarrow [2]T$
4:     if $m_i = 1$ then
5:         $f \leftarrow f \cdot \frac{\ell_{T,Q}(P)}{\nu_{T+Q}(P)}$, $T \leftarrow T + Q$
6:     elif $m_i = -1$ then
7:         $f \leftarrow f \cdot \frac{\ell_{T,-Q}(P)}{\nu_{T-Q}(P)}$, $T \leftarrow T - Q$
8:     end if
9: end for
10: return $f$
```

3 Elliptic Curves with Embedding Degrees 10 and 14

The construction of pairing-friendly curves necessitates special methods to ensure a small embedding degree k, which is crucial for efficient pairing computation. In addition, we also expect pairing-friendly curves have j-invariant 0 or 1728 such that they are equipped with efficiently computable endomorphisms and efficient formulas for point operation. Using the Brezing-Weng method [12], Freeman, Scott and Teske [24, Sect. 6] constructed a list of such curves with embedding degrees 10 and 14, which are summarized in Table 1. The formulas for optimal pairing on these curves are given in Table 2. It is straightforward to see that the number of iterations in Miller's algorithm on these curves is approximately $\log r/\varphi(k)$.

Remark 1. Using Eq. (2), the formula for the optimal pairing for the Cyclo(6.5)-10 family is expressed as $\left(f^{p}_{z^2,Q}(P)\right)^{(p^{10}-1)/r}$. Since a non-degenerate power of a pairing remains a pairing, we can replace $f^{p}_{z^2,Q}(P)$ by $f_{z^2,Q}(P)$ to save one Frobenius map.

Table 1. Important parameters for pairing-friendly curves with embedding degrees 10 and 14 from [24, Sect. 6].

family	k	$j(E)$	p	r	t
Cyclo(6.3)	10	1728	$\frac{1}{4}(z^{14}-2z^{12}+z^{10}+z^4+2z^2+1)$	$\Phi_{20}(z)$	z^2+1
Cyclo(6.5)	10	1728	$\frac{1}{4}(z^{12}-z^{10}+z^8-5z^6+5z^4-4z^2+4)$	$\Phi_{20}(z)$	$-z^6+z^4-z^2+2$
Cyclo(6.6)	10	0	$\frac{1}{3}(z^3-1)^2(z^{10}-z^5+1)+z^3$	$\Phi_{30}(z)$	z^3+1
Cyclo(6.3)	14	1728	$\frac{1}{4}(z^{18}-2z^{16}+z^{14}+z^4+2z^2+1)$	$\Phi_{28}(z)$	z^2+1
Cyclo(6.6)	14	0	$\frac{1}{3}(z-1)^2(z^{14}-z^7+1)+z^{15}$	$\Phi_{42}(z)$	z^8-z+1

3.1 New Formulas for Optimal Pairings on Target Curves

Recently, Dai, Zhang and Zhao [18] proposed a faster formula for pairing computation on the BW13-310 curve such that the length of the Miller loop can be reduced to around $\log r/(2\varphi(k))$. In this subsection, we show how to generalize this technique to our target curves. On this basis, we further propose an improved algorithm to reduce the performance penalty introduced by this new technique.

By the fact that the endomorphism ring of ordinary elliptic curves is commutative, we find that $\tau(Q) \in \mathbb{G}_2$ for any $Q \in \mathbb{G}_2$ as

$$\pi \circ \tau(Q) = \tau \circ \pi(Q) = \tau([p]Q) = [p]\tau(Q) \text{ and } [r]\tau(Q) = \tau([r]Q) = \mathcal{O}_E.$$

Furthermore, since the group order of $\mathbb{G}_2$ is prime, the endomorphism τ acts on $\mathbb{G}_2$ as a scalar, which is denoted as λ_2. In detail, we can fix the parameter of τ such that

$$\lambda_2 = \begin{cases} -z^{k/2}, & \text{in the Cyclo(6.3)-10, 14 and Cyclo(6.5)-10 families;} \\ z^k, & \text{in the Cyclo(6.6)-10 family;} \\ -z^k-1, & \text{in the Cyclo(6.6)-14 family.} \end{cases} \quad (3)$$

Table 2. Original formulas for optimal pairings on pairing-friendly curves with embedding degrees 10 and 14.

family-k	short vector	optimal pairing
Cyclo(6.3)-10	$[z^2, -1, 0, 0]$	$\left(f_{z^2,Q}(P)\right)^{(p^{10}-1)/r}$
Cyclo(6.5)-10	$[-1, z^2, 0, 0]$	$\left(f_{z^2,Q}(P)\right)^{(p^{10}-1)/r}$
Cyclo(6.6)-10	$[z, 0, -1, z^2]$	$\left(f_{z,Q}(P) \cdot f^{p^3}_{z^2,Q}(P) \cdot \ell_{\pi^7(Q),\pi^3([z^2]Q)}(P)\right)^{(p^{10}-1)/r}$
Cyclo(6.3)-14	$[z^2, -1, 0, 0, 0, 0]$	$\left(f_{z^2,Q}(P)\right)^{(p^{14}-1)/r}$
Cyclo(6.6)-14	$[z^2, z, 1, 0, 0, 0]$	$\left(f_{z^2,Q}(P) \cdot f^{p}_{z,Q}(P) \cdot \ell_{\pi^2(Q),\pi([z]Q)}(P)\right)^{(p^{14}-1)/r}$

By combining the two endomorphisms π and τ, we fortunately find that $\pi^m \circ \tau(Q) = [z]Q$ for any $Q \in \mathbb{G}_2$, where

$$m = \begin{cases} (k+2)/4, & \text{in the Cyclo(6.3)-10 and Cyclo(6.3)-14 families;} \\ 7, & \text{in the Cyclo(6.5)-10 and Cyclo(6.6)-10 families;} \\ 1, & \text{in the Cyclo(6.6)-14 family.} \end{cases}$$

This observation enables us to rewrite the formulas for optimal pairings on our target curves such that the number of Miller iterations can be reduced to around $\log r/(2\varphi(k))$, which is summarized in Lemma 1 below.

Lemma 1. *Let notation be as above. Then* $f_{z^2,Q} = f^{z}_{z,Q} \cdot f^{p^m}_{z,Q} \circ \hat{\tau}$, *where* $\hat{\tau}$ *is the dual of* τ.

Proof. It can obtained from [35, Lemma 3.5] that

$$f_{z^2,Q} = f^{z}_{z,Q} \cdot f_{z,[z]Q}. \tag{4}$$

Since $\pi^m \circ \tau(Q) = [z]Q$, it follows from [53, Theorem 1] and [17, Theorem 1] that

$$f_{z,[z]Q} = f_{z,\pi^m \circ \tau(Q)} = f^{p^m}_{z,\tau(Q)} = f^{p^m}_{z,Q} \circ \hat{\tau}^{p^m} = f^{p^m}_{z,Q} \circ \hat{\tau}. \tag{5}$$

Inserting Eq. (5) into Eq. (4), we have

$$f_{z^2,Q} = f^{z}_{z,Q} \cdot f^{p^m}_{z,Q} \circ \hat{\tau},$$

which completes the proof. □

Based on Lemma 1, we can derive new formulas for optimal pairings on our target curves by executing the following two steps:

-Step 1. We first replace $f_{z^2,Q}(P)$ by $f^{z}_{z,Q}(P) \cdot f^{p^m}_{z,Q}(\hat{\tau}(P))$ in the original formulas for optimal pairings. In particular, we can also replace the point $[z]Q$ by $\pi^m \circ \tau(Q)$ at the final line in the Cyclo(6.6)-10 and Cyclo(6.6)-14 families.

Table 3. Optimized formulas for optimal pairings on pairing-friendly curves with embedding degrees 10 and 14.

family-k	optimal pairing
Cyclo(6.3)-10	$\left(f_{z,Q}^{z\cdot p^7}(P)\cdot f_{z,Q}(\hat{\tau}(P))\right)^{(p^{10}-1)/r}$
Cyclo(6.5)-10	$\left(f_{z,Q}^{z\cdot p^3}(P)\cdot f_{z,Q}(\hat{\tau}(P))\right)^{(p^{10}-1)/r}$
Cyclo(6.6)-10	$\left(f_{z,Q}^{1+z\cdot p^3}(P)\cdot f_{z,Q}(\hat{\tau}(P))\cdot(y_P-y_Q)^{p^7}\right)^{(p^{10}-1)/r}$
Cyclo(6.3)-14	$\left(f_{z,Q}^{z\cdot p^{10}}(P)\cdot f_{z,Q}(\hat{\tau}(P))\right)^{(p^{14}-1)/r}$
Cyclo(6.6)-14	$\left(f_{z,Q}^{1+z\cdot p^{13}}(P)\cdot f_{z,Q}(\hat{\tau}(P))\cdot(y_P-y_Q)^{p}\right)^{(p^{14}-1)/r}$

Table 4. Parameters of the pairing-friendly curves with embedding degrees 10 and 14 at the updated 128-bit security level.

curve	family-k	seed z	r bits	p bits	p^k bits	DL cost in $\mathbb{F}_{p^k}$
BW10-480	Cyclo(6.5)-10	$2^5+2^{14}+2^{15}+2^{18}+2^{36}+2^{40}$	321	480	4791	128
BW10-511	Cyclo(6.6)-10	$2^7+2^{13}+2^{26}-2^{32}$	256	511	5101	150
BW10-512	Cyclo(6.3)-10	$1+2^3+2^{17}+2^{32}+2^{35}+2^{36}$	294	512	5111	129
BW14-351	Cyclo(6.6)-14	$2^6-2^{12}-2^{14}-2^{22}$	265	351	4908	149
BW14-382	Cyclo(6.3)-14	$1+2^{10}+2^{13}-2^{16}+2^{19}+2^{21}$	256	382	5338	129

-Step 2. Utilizing the property that a non-degenerate power of a pairing remains a pairing, we then can raise the output of the Miller loop to a p^{k-m}-power such that the exponent of the second Miller function is equal to 1.

The new formulas for the optimal pairing for our selected curves are summarized in Table 3. Clearly, the most costly part of the Miller loop is to compute $f_{z,Q}^{z\cdot p^{k-m}}(P)\cdot f_{z,Q}(\hat{\tau}(P))$, enabling the execution of Miller's algorithm in around $\log|z|$ iterations within the same loop, albeit with a slightly higher computational cost per iteration. However, compared to the original formulas, the new ones require an additional exponentiation by z. Fortunately, the exponentiation can be integrated with the computation of $f_{z,Q}(\hat{\tau}(P))$ to share several squarings. Specifically, we first calculate $f_{z,Q}(P)$ and store all line function evaluations required for computing $f_{z,Q}(\hat{\tau}(P))$ at the first loop. Subsequently, given the initial value $f_{z,Q}^{p^{k-m}}(P)$, we then compute $f_{z,Q}^{z\cdot p^{k-m}}(P)\cdot f_{z,Q}(\hat{\tau}(P))$ at the second loop. The optimized procedure for computing this value is presented in Algorithm 2. Thanks to the final exponentiation, the value of g^{-1} can be replaced by $\bar{g}$ in Line 10 of Algorithm 2, where $\bar{g}$ represents the conjugate of g.

3.2 Choice of Parameters at the 128-Bit Security Level

The choice of parameters of pairing-friendly curves should be careful for achieving high performance implementation at the desired security level. In this paper, we focus on the performance of pairing computation at the 128-bit security level. To this end, the size of full extension field $\mathbb{F}_{p^k}$ should be large enough to

Algorithm 2: Computing $f_{z,Q}^{z\cdot p^{k-m}}(P)\cdot f_{z,Q}(\hat{\tau}(P))$

```
**Input:** $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_2$, $z = \sum_{i=0}^{L} z_i \cdot 2^i$ with $z_i \in \{-1,0,1\}$
**Output:** $f_{z,Q}^{z\cdot p^{k-m}}(P)\cdot f_{z,Q}(\hat{\tau}(P))$
1: $T \leftarrow Q$, $f \leftarrow 1$, tab$\leftarrow [\,]$, $j \leftarrow 0$
2: for $i = L-1$ down to $0$ do
3:     $f \leftarrow f^2 \cdot \ell_{T,T}(P)$, tab$[j] \leftarrow \ell_{T,T}(\hat{\tau}(P))$, $T \longleftarrow 2T$, $j \leftarrow j+1$    // SDBL
4:     if $z_i = 1$ then
5:         $f \leftarrow f \cdot \ell_{T,Q}(P)$,tab$[j] \leftarrow \ell_{T,Q}(\hat{\tau}(P))$, $T \leftarrow T+Q$, $j \leftarrow j+1$    // SADD
6:     elif $z_i = -1$ then
7:         $f \leftarrow f \cdot \ell_{T,-Q}(P)$, tab$[j] \leftarrow \ell_{T,-Q}(\hat{\tau}(P))$, $T \leftarrow T-Q$, $j \leftarrow j+1$    // SADD
8:     end if
9: end for
10: $g \leftarrow f^{p^{k-m}}$, $h \leftarrow g$, $j \leftarrow 0$
11: for $i = L-1$ down to $0$ do
12:     $h \leftarrow h^2 \cdot$tab$[j]$, $j \leftarrow j+1$
13:     if $z_i = 1$ then
14:         $h \leftarrow h \cdot g \cdot$tab$[j]$, $j \leftarrow j+1$
15:     elif $z_i = -1$ then
16:         $h \leftarrow h \cdot \bar{g} \cdot$tab$[j]$, $j \leftarrow j+1$
17:     end if
18: end for
19: return $h$
```

resist attacks from the variants of NFS. The concrete security can be estimated using the source code provided by Guillevic and Singh [33], which is available at https://gitlab.inria.fr/tnfs-alpha/alpha/tree/master/sage.

On this basis, to maximize the efficiency of pairing computation, we also expect

(a) the selected prime p satisfies that $p \equiv 1 \bmod k$;
(b) the sum of bit length and Hamming weight (in non-adjacent form) of the selected seed z is as small as possible.

In more detail, the condition (a) ensures that the full extension field $\mathbb{F}_{p^k}$ can be represented as $\mathbb{F}_p[v]/(v^k - \alpha)$ for some $\alpha \in \mathbb{F}_p^*$ [40, Theorem 3.75], which actually can be constructed as a tower of quadratic and $k/2$-th extensions:

$$\mathbb{F}_p \xrightarrow{\xi^{k/2}-\alpha} \mathbb{F}_{p^{k/2}} \xrightarrow{v^2-\xi} \mathbb{F}_{p^k}.$$

This construction induces fast multiplication and squaring arithmetic operations in $\mathbb{F}_{p^k}$; the condition (b) aims to minimize the number of Miller iterations in **Alg**. 2. In fact, the computation of the final exponentiation also benefits from condition (b) since this step consists of a large number of exponentiations by z (see Sect. 4.3). Table 4 summarizes our chosen seeds z under the above conditions, together with the corresponding sizes of the curve parameters. Notably, while Guillevic [31, Table 6] selected a seed for the Cyclo(6.6)-14 family, this seed fails to meet the condition (a).

Curve Name: For convenience, all the candidate curves listed in Table 4 are collectively called BW curves since they are essentially generated using the Brezing-Weng method. Moreover, we distinguish each curve by its embedding degree and the size of the characteristic p. For instance, the BW14-351 curve is referred to as the curve constructed from the Cyclo(6.6)-14 family defined over a 351-bit prime field.

4 Pairing Computation

In this section, we first describe explicit formulas for Miller doubling and addition steps. In particular, the technique of lazy reduction [6,45] has been fully exploited to reduce the number of modular reductions required in Miller's algorithm. Then, we show how to perform the final exponentiation efficiently. Finally, we present detailed operation counts for pairing computation on different curves.

Notations. The cyclotomic group $\mathbb{G}_{\Phi_k(p)}$ is defined by $\mathbb{G}_{\Phi_k(p)} = \{a \in \mathbb{F}_{p^k} \mid a^{\Phi_k(p)} = 1\}$. The notation $\times$ is used to denote field multiplication without reduction. We use the following notation to denote the cost of operations:(i) $\mathbf{a}$, $\mathbf{m}$, $\mathbf{m}_u$, $\mathbf{s}$, $\mathbf{s}_u$, $\mathbf{i}$ and $\mathbf{r}$ denote the cost of addition, multiplication, multiplication without reduction, squaring, squaring without reduction, inversion and modular reduction in $\mathbb{F}_p$, respectively; (ii) $\tilde{\mathbf{a}}$, $\tilde{\mathbf{m}}$, $\tilde{\mathbf{m}}_u$, $\tilde{\mathbf{m}}_\xi$, $\tilde{\mathbf{s}}$, $\tilde{\mathbf{s}}_u$, $\tilde{\mathbf{i}}$ and $\tilde{\mathbf{r}}$ represent the cost of addition, multiplication, multiplication without reduction, multiplication by ξ, squaring, squaring without reduction, inversion and modular reduction in $\mathbb{F}_{p^{k/2}}$, respectively; (iii) $\mathbf{M}$, $\mathbf{S}$, $\mathbf{f}$ and $\mathbf{I}$ represent the cost of multiplication, squaring, Frobenius map and inversion in $\mathbb{F}_{p^k}$, respectively; (iv) $\mathbf{S}_c$ and $\mathbf{e}$ represent the cost of squaring and exponentiation by z in the cyclotomic group $\mathbb{G}_{\Phi_k(p)}$, respectively.

4.1 Miller Loop on Curves of Form $y^2 = x^3 + b$

Let $E : y^2 = x^3 + b$ be a curve over $\mathbb{F}_p$ with embedding degree 10 or 14. Then E admits a quadratic twist $E' : y^2 = x^3 + b/\xi^3$ over $\mathbb{F}_{p^{k/2}}$. The associated untwisting isomorphism from E' to E is given by

$$\phi : (x, y) \to (x\xi, y\xi v).$$

To avoid field inversions when performing point operations, points can be represented in projective coordinates. For this curve shape, it is convenient to use Jacobian coordinates, that is, an affine point (x, y) corresponds to a triplet (X, Y, Z) with $x = X/Z^2$ and $y = Y/Z^3$.

Shared Doubling Step (SDBL). Let $T = (X, Y, Z) \in E'(\mathbb{F}_{p^{k/2}})[r]$ be in Jacobian coordinates. The formula for computing the doubling point $[2]T = (X_3, Y_3, Z_3)$ is derived from [7, Sect. 4.3], where

$$X_3 = X(\frac{9}{4}X^3 - 2Y^2), Y_3 = \frac{9}{4}X^3(2Y^2 - \frac{3}{2}X^3) - Y^4, Z_3 = YZ.$$

By the form of the untwisting map ϕ, the image point $\phi(T) \in \mathbb{G}_2$ can be represented as $(X\xi, Y\xi v, Z)$. Thanks to the technique of denominator elimination, the line function $l_{\phi(T),\phi(T)}$ evaluated at $P = (x_P, y_P)$ and $\hat{\tau}(P) = (\tilde{x}_P, y_P)$ can be simplified as

$$\begin{aligned} l_{\phi(T),\phi(T)}(P) &= 2Z_3Z^2y_P + \big((3X^3 - 2Y^2)\cdot\xi - 3X^2Z^2x_P\big)v,\\ l_{\phi(T),\phi(T)}(\hat{\tau}(P)) &= 2Z_3Z^2y_P + \big((3X^3 - 2Y^2)\cdot\xi - 3X^2Z^2\tilde{x}_P\big)v. \end{aligned}$$

It is evident that the two line evaluations share a large amount of variables. In addition, the technique of lazy reduction can be employed when computing Y_3. Thus, we can obtain the above two line evaluations using the following sequence of operations:

$$\begin{aligned} &A = 3X^2, B = A\cdot X, C = \frac{B}{2}, D = C + \frac{C}{2}, E = Y^2, F = 2E, U_0 = D\times(F - C),\\ &U_1 = E\times E,\quad Y_3 = (U_0 - U_1) \bmod p, X_3 = X\cdot(D - F), Z_3 = Y\cdot Z, G = Z^2,\\ &I = G\cdot Z_3\cdot(2y_P), J = A\cdot G, K = (B - F)\cdot\xi, L = J\cdot x_P, M = J\cdot\tilde{x}_P,\\ &l_{\phi(T),\phi(T)}(P) = I + (K - L)v, l_{\phi(T),\phi(T)}(\hat{\tau}(P)) = I + (K - M)v. \end{aligned}$$

The total operation count for point doubling and two line evaluations is $5\tilde{\mathbf{m}} + \tilde{\mathbf{m}}_u + \tilde{\mathbf{s}}_u + \tilde{\mathbf{m}}_\xi + 3\tilde{\mathbf{s}} + \frac{3k}{2}\mathbf{m} + \tilde{\mathbf{r}} + 13\tilde{\mathbf{a}} + \mathbf{a}$, assuming that the computation of division by 2 and $U_0 - U_1$ requires one and two additions, respectively.

Shared Addition Step (SADD). Let $T = (X, Y, Z), Q = (X_2, Y_2, Z_2) \in E'(\mathbb{F}_{p^{k/2}})[r]$ be in Jacobian coordinates with $Z \neq 0$, $Z_2 = 1$ and $T \neq Q$. Then one can compute the point $T + Q = (X_3, Y_3, Z_3)$ using the mixed addition formula [7, **Sect**. 4.3], which is given as

$$\theta = Y_2Z^3 - Y, \beta = X_2Z^2 - X, X_3 = \theta^2 - 2X\beta^2 - \beta^3, Y_3 = \theta(X\beta^2 - X_3) - Y\beta^3, Z_3 = Z\beta.$$

Subsequently, the line function $l_{\phi(T),\phi(Q)}$ evaluated at P and $\hat{\tau}(P)$ can be expressed as

$$\begin{aligned} l_{\phi(T),\phi(Q)}(P) &= Z_3y_p + \big((\theta X_2 - Y_2Z_3)\cdot\xi - \theta x_P\big)v,\\ l_{\phi(T),\phi(Q)}(\hat{\tau}(P)) &= Z_3y_p + \big((\theta X_2 - Y_2Z_3)\cdot\xi - \theta\tilde{x}_P\big)v. \end{aligned}$$

Again, by taking advantage of the technique of lazy reduction, we perform the following sequence of operations to compute the above point addition and two line evaluations, which costs $6\tilde{\mathbf{m}} + 4\tilde{\mathbf{m}}_u + \tilde{\mathbf{m}}_\xi + 3\tilde{\mathbf{s}} + \frac{3k}{2}\mathbf{m} + 2\tilde{\mathbf{r}} + 12\tilde{\mathbf{a}}$:

$$\begin{aligned}
&A = Z^2, \theta = Y_2 \cdot A \cdot Z - Y, \beta = X_2 \cdot A - X, B = \beta^2, C = \beta \cdot B, D = X \cdot B,\\
&X_3 = \theta^2 - 2D - C, U_0 = \theta \times (D - X_3), U_1 = Y \times C, Y_3 = (U_0 - U_1) \bmod p,\\
&Z_3 = Z \cdot \beta, E = Z_3 \cdot y_P, F = \theta \cdot x_P, G = \theta \cdot \tilde{x}_P, U_2 = \theta \times X_2, U_3 = Y_2 \times Z_3,\\
&H = (U_2 - U_3) \bmod p, I = H \cdot \xi, l_{\phi(T),\phi(Q)}(P) = E + (I - F)v,\\
&l_{\phi(T),\phi(Q)}(\hat{\tau}(P)) = E + (I - G)v.
\end{aligned}$$

4.2 Miller Loop on Curves of Form $y^2 = x^3 + ax$

Let $E : y^2 = x^3 + ax$ be a curve over $\mathbb{F}_p$ with embedding degree 10 or 14. Then E admits a quadratic twist $E' : y^2 = x^3 + a'$ over $\mathbb{F}_{p^{k/2}}$, where $a' = a \cdot \xi^2$. As a consequence, the associated untwisting isomorphism from E' to E can be expressed as

$$\phi : (x, y) \to (x, y) \to (x/\xi, y/(\xi v)).$$

For this curve shape, we represent an affine point (x, y) in the weight-$(1, 2)$ coordinates (X, Y, Z) satisfying that $x = X/Z$ and $y = Y/Z^2$ [16, **Sect**. 4].

Shared Doubling Step (SDBL). Let $T = (X, Y, Z) \in E'(\mathbb{F}_{p^{k/2}})[r]$ be in weight-$(1, 2)$ coordinates. For this curve shape, the point doubling formula for computing $[2]T = (X_3, Y_3, Z_3)$ is derived from [16, **Sect**. 4], which is expressed as

$$X_3 = (X^2 - a'Z^2)^2, Y_3 = 2Y(X^2 - a'Z^2)\big(2(X^2 + a'Z^2)^2 - X_3\big), Z_3 = 4Y^2.$$

In this case, it is more convenient to perform line evaluations on the twisted curve. In other words, we compute the line function $l_{T,T}$ evaluated at $\phi^{-1}(P) = (x_P\xi, y_P\xi v)$ and $\phi^{-1} \circ \hat{\tau}(P) = (-x_P\xi, \tilde{y}_P\xi v)$. The explicit formulas are given by

$$\begin{aligned}
l_{T,T}(\phi^{-1}(P)) &= (X^3 - a'XZ^2) - (3X^2Z + a'Z^3)x_P\xi + 2YZy_P\xi v,\\
l_{T,T}(\phi^{-1} \circ \hat{\tau}(P)) &= (X^3 - a'XZ^2) + (3X^2Z + a'Z^3)x_P\xi + 2YZ\tilde{y}_P\xi v.
\end{aligned}$$

Accordingly, the point doubling and two line evaluations can be accomplished by performing the following sequences of operations at a cost of $5\tilde{\mathbf{m}} + 5\tilde{\mathbf{s}} + \frac{3k}{2}\mathbf{m} + 2\tilde{\mathbf{m}}_\xi + \tilde{\mathbf{m}}_{a'} + 9\tilde{\mathbf{a}}$ ($\tilde{\mathbf{m}}_{a'}$ denotes the cost of multiplication by a'):

$$\begin{aligned}
&A = X^2, B = 2Y, C = a' \cdot Z^2, D = A - C, E = A + C, X_3 = D^2, Z_3 = B^2, F = B \cdot Z,\\
&Y_3 = B \cdot D \cdot (2E^2 - X_3), G = F \cdot \xi, I = X \cdot D, H = (2A + E) \cdot Z \cdot x_P, J = y_P \cdot G,\\
&\tilde{J} = \tilde{y}_P \cdot G, K = H \cdot \xi, l_{T,T}(\phi^{-1}(P)) = I - K + Jv, l_{T,T}(\phi^{-1} \circ \hat{\tau}(P)) = I + K + \tilde{J}v.
\end{aligned}$$

Shared Addition Step (SADD). Let $T = (X, Y, Z), Q = (X_2, Y_2, Z_2) \in E'(\mathbb{F}_{p^{k/2}})[r]$ be in weight-(1, 2) coordinates with $Z \neq 0$, $Z_2 = 1$ and $T \neq Q$.

We adopt the mixed addition formula [16, **Sect**. 4] to compute the point $T+Q=(X_3, Y_3, Z_3)$, which is given by

$$U = X - X_2Z, S = U^2Z, X_3 = (Y - Y_2Z^2)^2 - (X + X_2Z)S,$$
$$Y_3 = \big((Y - Y_2Z^2)(XS - X_3) - YSU\big)UZ, Z_3 = (UZ)^2.$$

Subsequently, the line function $l_{T,Q}$ evaluated at $\phi^{-1}(P)$ and $\phi^{-1} \circ \hat{\tau}(P)$ are given by

$$l_{T,Q)}(\phi^{-1}(P)) = \big((Y - Y_2Z^2)X_2 - UZY_2\big) - (Y - Y_2Z^2)\xi x_P + y_PUZ\xi v,$$
$$l_{T,Q}(\phi^{-1} \circ \hat{\tau}(P)) = \big((Y - Y_2Z^2)X_2 - UZY_2\big) + (Y - Y_2Z^2)\xi x_P + \tilde{y}_PUZ\xi v.$$

The following sequence of operations can be used for computing the above mixed point addition and two line evaluations at a cost of $6\tilde{\mathbf{m}} + 6\tilde{\mathbf{m}}_u + 2\tilde{\mathbf{m}}_\xi + 3\tilde{\mathbf{s}} + \frac{3k}{2}\mathbf{m} + 3\tilde{\mathbf{r}} + 10\tilde{\mathbf{a}}$:

$$A = Z^2, B = X_2 \cdot Z, C = Y_2 \cdot A, D = X - B, E = Y - C, F = Z \cdot D, G = F \cdot D,$$
$$X_3 = \big(E \times E - (X + B) \times G\big) \bmod p, H = X \cdot G - X_3, I = E \cdot F, J = G^2,$$
$$Y_3 = (I \times H - Y \times J) \bmod p, Z_3 = F^2, K = (E \times X_2 - F \times Y_2) \bmod p, L = E \cdot x_P \cdot \xi,$$
$$M = F \cdot \xi, N = M \cdot y_P, \tilde{N} = M \cdot \tilde{y}_P, l_{T,Q}(\phi^{-1}(P)) = (K - L) + Nv,$$
$$l_{T,Q}(\phi^{-1} \circ \hat{\tau}(P)) = (K + L) + \tilde{N}v.$$

4.3 The Final Exponentiation

The final exponentiation is the other time-consuming stage of pairing computation. This step aims to raise the output of the Miller loop to the power of $(p^k - 1)/r$. For our target curves, the large exponent can be split into two parts:

$$(p^{k-1})/r = \underbrace{(p+1)(p^{k/2} - 1)}_{\text{easy part}} \cdot \underbrace{\Phi_k(p)/r}_{\text{hard part}} \cdot$$

The exponentiation to the power of the easy part yields an element $f \in \mathbb{G}_{\Phi_k(p)}$, which costs only $\mathbf{I} + 3\mathbf{M} + 2\mathbf{f}$. The major bottleneck of the final exponentiation arises from the exponentiation to the power of the hard part. Observing that a non-degenerate power of a pairing remains a pairing, Fuentes-Castañeda *et al*. [25] proved that it suffices to raise f to the power of a multiple h of $\Phi_k(p)/r$, where h can be written in the base p as

$$h = h_0 + h_1 \cdot p + \cdots + h_{\varphi(k)-1} \cdot p^{\varphi(k)-1}.$$

As a consequence, the LLL algorithm is applied to obtain small coefficients h_i. In essence, this method aims to minimize the number of iterations required for the final exponentiation. Nevertheless, it may still be challenging to devise an optimized routine of the $\varphi(k)$ small exponentiations f^{h_i}. For example, when applying this method to the BW14-351 curve, the six coefficients h_i are given as follows:

$$\begin{aligned}
h_0 =& z^{13} + z^{12} + z^{11} - z^6 + 3z^5 + z^3,\\
h_1 =& - z^{13} - z^{12} - 2z^{11} - z^{10} - z^9 + z^6 - 2z^5 + z^4 - 3z^3,\\
h_2 =&(1 + z^3)(z^{10} + z^9 + z^8) - z^6 + 2z^5 - z^4 - z^3 + 2z^2 - z,\\
h_3 =& - z^{13} - z^{12} - z^{11} + z^6 - 2z^5 + z^4 + z^2 + z + 1,\\
h_4 =& z^{13} + z^{12} + z^{11} - z^8 - z^7 - 2z^6 + 2z^5 - z^4 - 3,\\
h_5 =&z^{14} - z^{11} + 4z^6 - 2z^5 + z^4.
\end{aligned}$$

Thus, the cost of computing f^{h_i} consists of 14 exponentiations by z and a large amount of full extension field multiplications.

Based on the fact that $f^{\Phi_k(p)} = 1$, we can further substitute the exponent h with $\lambda = h+\delta\Phi_k(p)$ for some integer δ. In particular, since $\Phi_k(p) = \sum_{i=0}^{\varphi(k)}(-1)^i p^i$ in our case, the new exponent λ can be written in the base of p as

$$\lambda = \lambda_0 + \lambda_1 \cdot p + \cdots + \lambda_{\varphi(k)} \cdot p^{\varphi(k)},$$

where $\lambda_i = h_i + (-1)^i\delta$ for $i \in \{0, 1, \cdots, \varphi(k) - 1\}$ and $\lambda_{\varphi(k)} = \delta$. Therefore, the careful selection of the parameter δ may facilitate faster final exponentiation. We now revisit the final exponentiation on the BW14-351 curve. By setting $\lambda_6 = -(z^{13} + z^{12} + z^{11} + 3z^5) + (z^6 + z^5 + z^4)$, we have

$$\begin{aligned}
&\lambda_0 = h_0 + \lambda_6 = z^5 + z^4 + z^3, && \lambda_1 = h_1 - \lambda_6 = -z^{11} - z^{10} - z^9 - 3z^3,\\
&\lambda_2 = h_2 + \lambda_6 = z^{10} + z^9 + z^8 - z^3 + 2z^2 - z, && \lambda_3 = h_3 - \lambda_6 = z^2 + z + 1,\\
&\lambda_4 = h_4 + \lambda_6 = -z^8 - z^7 - z^6 - 3, && \lambda_5 = h_5 - \lambda_6 = z^{14} + z^{13} + z^{12} + 3z^6.
\end{aligned}$$

It is straightforward to see that the six coefficients λ_i satisfy the following relations:

$$\begin{aligned}
&\lambda_3 = z^2 + z + 1, && \lambda_0 = z^3\lambda_3, && \lambda_4 = -(z^3\lambda_0 + 3), \lambda_2 = -(z^2\lambda_4 + z\lambda_3),\\
&\lambda_1 = z^3\lambda_4, && \lambda_6 = z^2\lambda_1 + z\lambda_0, && \lambda_5 = -z^3\lambda_1.
\end{aligned}$$

In conclusion, the hard part exponentiation on the BW14-351 curve benefits from the easy relation between λ_i. In Table 5, we list our selected coefficients $\lambda_0, \lambda_1, \cdots, \lambda_{\varphi(k)}$ and the corresponding sequence of operations on the five candidate curves.

4.4 Computational Cost

The construction of tower fields and the curve equations for the five candidate pairing-friendly curves are presented in Table 6. We now discuss the operation counts of pairing computation on these curves. To this aim, we first count the number of finite field arithmetic operations. For the Frobenius map and the

Table 5. The exponentiation of the hard part on the five candidate pairing-friendly curves. The values of f^3 and f^4 can be obtained during the computation of f^z. Therefore, we assume that the computation of f^3 requires one multiplication, while the computation of f^4 is free. The notation $\bar{f}_i$ is denoted as the conjugate of f_i.

BW10-480	$\lambda_0 = z^8 - 4z^2, \lambda_1 = z^{10} - z^8 - 4z^4 + 4z^2, \lambda_2 = z^6 - z^4 - 4, \lambda_3 = -z^6 + 4, \lambda_4 = 0$
	Input:$f \in \mathbb{G}_{\Phi_{14}(p)}$, **Output**:$h \in \mathbb{G}_T$, **Cost**:$10\mathbf{e} + 6\mathbf{M} + 3\mathbf{f}$ $f_1 \leftarrow f^{z^4}, f_2 \leftarrow f_1^{z^2} \cdot \bar{f}^4, f_3 \leftarrow f_2^{z^2}, f_4 \leftarrow f_3^{z^2}, f_5 \leftarrow f_2 \cdot \bar{f}_1, f_6 \leftarrow f_4 \cdot \bar{f}_3,$ $h \leftarrow f_3 \cdot f_6^p \cdot f_5^{p^2} \cdot \bar{f}_2{}^p$
BW10-511	$\lambda_0 = -z^{13} + 2z^{10} - z^7 - 3, \lambda_1 = -z^{10} + 2z^7 - z^4, \lambda_2 = -z^7 + 2z^4 - z,$ $\lambda_3 = (z^{14} - 2z^{11} + z^8 + 3z) - (z^9 - 2z^6 + z^3), \lambda_4 = (z^{11} - 2z^8 + z^5) - (z^6 - 2z^3 + 1)$
	Input:$f \in \mathbb{G}_{\Phi_{10}(p)}$, **Output**:$h \in \mathbb{G}_T$, **Cost**:$14\mathbf{e} + 9\mathbf{M} + \mathbf{S}_c + 4\mathbf{f}$ $f_1 \leftarrow f^{z^6 - 2z^3 + 1}, f_2 \leftarrow f_1^z, f_3 \leftarrow f_2^{z^2}, f_4 \leftarrow f_3^z, f_5 \leftarrow f_4^z, f_6 \leftarrow f_5^{z^2} \cdot f^3,$ $f_7 \leftarrow f_6^z \cdot \bar{f}_3, f_8 \leftarrow f_5 \cdot \bar{f}_1, h \leftarrow \bar{f}_6 \cdot \bar{f}_4{}^p \cdot \bar{f}_2{}^{p^2} \cdot f_7{}^{p^3} \cdot f_8{}^{p^4}$
BW10-512	$\lambda_0 = z^6 - 2z^4 + z^2, \lambda_1 = z^4 - 2z^2 + 1, \lambda_2 = -z^{12} + 2z^{10} - z^8 - 4,$ $\lambda_3 = -z^{10} + 2z^8 - z^6, \lambda_4 = -z^8 + 2z^6 - z^4$
	Input:$f \in \mathbb{G}_{\Phi_{10}(p)}$, **Output**:$h \in \mathbb{G}_T$, **Cost**:$12\mathbf{e} + 7\mathbf{M} + \mathbf{S}_c + 4\mathbf{f}$ $f_1 \leftarrow f^{z^4 - 2z^2 + 1}, f_2 \leftarrow f_1^{z^2}, f_3 \leftarrow f_2^{z^2}, f_4 \leftarrow f_3^{z^2}, f_5 \leftarrow f_4^{z^2} \cdot f^4,$ $h \leftarrow f_2 \cdot f_1^p \cdot \bar{f}_5{}^{p^2} \cdot \bar{f}_4{}^{p^3} \cdot \bar{f}_3{}^{p^4}$
BW14-351	**Input**:$f \in \mathbb{G}_{\Phi_{14}(p)}$, **Output**:$h \in \mathbb{G}_T$, **Cost**:$14\mathbf{e} + 12\mathbf{M} + 6\mathbf{f}$ $f_1 \leftarrow f^{z^2 + z + 1}, f_2 \leftarrow f_1^z, f_3 \leftarrow f_2^{z^2}, f_4 \leftarrow f_3^z, f_5 \leftarrow f_4^{z^2} \cdot f^3 f_5 \leftarrow \bar{f}_5, f_6 \leftarrow f_5^{z^2},$ $f_7 \leftarrow f_2 \cdot f_6, f_8 \leftarrow f_6^z, f_9 \leftarrow f_8^{z^2}, f_{10} \leftarrow f_4 \cdot f_9, f_{11} \leftarrow \bar{f}_9{}^z,$ $h \leftarrow f_3 \cdot \bar{f}_8{}^p \cdot \bar{f}_7{}^{p^2} \cdot f_1{}^{p^3} \cdot f_5{}^{p^4} \cdot f_{11}{}^{p^5} \cdot f_{10}{}^{p^6}$
BW14-382	$\lambda_0 = z^{10} - 2z^8 + z^6, \lambda_1 = z^8 - 2z^6 + z^4, \lambda_2 = z^6 - 2z^4 + z^2, \lambda_3 = z^4 - 2z^2 + 1,$ $\lambda_4 = -z^{16} + 2z^{14} - z^{12} - 4, \lambda_5 = -z^{14} + 2z^{12} - z^{10}, \lambda_6 = -z^{12} + 2z^{10} - z^8$
	Input:$f \in \mathbb{G}_{\Phi_{14}(p)}$, **Output**:$h \in \mathbb{G}_T$, **Cost**:$16\mathbf{e} + 7\mathbf{M} + \mathbf{S}_c + 4\mathbf{f}$ $f_1 \leftarrow f^{z^4 - 2z^2 + 1}, f_2 \leftarrow f_1^{z^2}, f_3 \leftarrow f_2^{z^2}, f_4 \leftarrow f_3^{z^2}, f_5 \leftarrow f_4 \cdot f_3^p \cdot f_2^{p^2},$ $f_6 \leftarrow (f_5^{z^6} \cdot f^4)^{p^4}, h \leftarrow f_1^{p^3} \cdot f_5 \cdot \bar{f}_6$

inversion operation, we adopt the formulas described in [32, **Sect**. 5]. For multiplication and squaring arithmetic, we combine the lazy reduction technique [6,45] and the Karatsuba algorithm [37]. In particular, cyclotomic squaring arithmetic can be accelerated using the formula described in [29, Sect. 2.1]. The exact operation counts for finite field arithmetic across different pairing-friendly curves are presented in Table 7.

Recall from **Sect**. 3.1 that the optimized formulas for the Miller function on our target curves are expressed as

$$\begin{cases} f_{z,Q}^{z \cdot p^{k-m}}(P) \cdot f_{z,Q}(\hat{\tau}(P)) \cdot f_{z,Q}(P) \cdot (y_P - y_Q)^{p^m}, & \text{if } j(E) = 1728; \\ f_{z,Q}^{z \cdot p^{k-m}}(P) \cdot f_{z,Q}(\hat{\tau}(P)), & \text{if } j(E) = 0. \end{cases}$$

Table 6. Parameters of full extension fields and curve equations for the five candidate pairing-friendly curves.

curve	full extension field	original curve E	twisted curve E'
BW10-480	$\mathbb{F}_p \xrightarrow{\xi^5+11} \mathbb{F}_{p^5} \xrightarrow{v^2-\xi} \mathbb{F}_{p^{10}}$	$y^2 = x^3 + x$	$y^2 = x^3 + \xi^2 x$
BW10-511	$\mathbb{F}_p \xrightarrow{\xi^5+4} \mathbb{F}_{p^5} \xrightarrow{v^2-\xi} \mathbb{F}_{p^{10}}$	$y^2 = x^3 - 2$	$y^2 = x^3 - 2/\xi^3$
BW10-512	$\mathbb{F}_p \xrightarrow{\xi^5+17} \mathbb{F}_{p^5} \xrightarrow{v^2-\xi} \mathbb{F}_{p^{10}}$	$y^2 = x^3 + x$	$y^2 = x^3 + \xi^2 x$
BW14-351	$\mathbb{F}_p \xrightarrow{\xi^7-2} \mathbb{F}_{p^7} \xrightarrow{v^2-\xi} \mathbb{F}_{p^{14}}$	$y^2 = x^3 + 3$	$y^2 = x^3 + 3/\xi^3$
BW14-382	$\mathbb{F}_p \xrightarrow{\xi^7-17} \mathbb{F}_{p^7} \xrightarrow{v^2-\xi} \mathbb{F}_{p^{14}}$	$y^2 = x^3 + x$	$y^2 = x^3 + \xi^2 x$

The computation of $f_{z,Q}^{z\cdot p^{k-m}}(P) \cdot f_{z,Q}(\hat{\tau}(P))$ can be performed using Algorithm 2, and it requires additional $2\mathbf{M} + \mathbf{f} + \tilde{\mathbf{a}}$ to complete the final step of the Miller iteration on curves with j-invariant 1728. In conclusion, the total operation count of Miller Loop (ML) is

$$\begin{aligned} ML = &\underbrace{2\mathbf{M} + \mathbf{f} + \tilde{\mathbf{a}}}_{\text{if} j(E)=1728} + \underbrace{(\texttt{nbits}(z) - 1) \cdot \texttt{SDBL} + (\texttt{hw}(z) - 1) \cdot \texttt{SADD}}_{\text{Lines 1-9 in Alg.2}} + \\ &\underbrace{\big((\texttt{nbits}(z) - 1) + 2\texttt{hw}(z) - 2\big) \cdot \mathbf{M} + \big(\texttt{nbits}(z) - 1\big) \cdot \mathbf{S} + \mathbf{f}}_{\text{Lines 10-16 in Alg.2}}, \end{aligned} \tag{6}$$

where $\texttt{nbits}(z)$ and $\texttt{hw}(z)$ represent the bit length and the Hamming weight in 2-non-adjacent form of the seed z, respectively. We use n_1, n_2, n_3 and n_4 to denote the number of $\mathbf{e}$, $\mathbf{M}$, $\mathbf{S}_c$ and $\mathbf{f}$ required for the exponentiation to the power of the hard part, respectively. Then the total operation count of the final exponentiation (FE) is

$$\begin{aligned} \text{FE} = &\underbrace{\mathbf{I}+3\mathbf{M}+2\mathbf{f}}_{\text{easy part}} + \underbrace{n_1\big((\texttt{nbits}(z)-1)\mathbf{S}_c + (\texttt{hw}(z)-1)\mathbf{M}\big) + n_2\mathbf{M} + n_3\mathbf{S}_c + n_4\mathbf{f}}_{\text{hard part}} \\ = &\mathbf{I} + \big(n_1(\texttt{hw}(z) - 1) + n_2 + 3\big)\mathbf{M} + \big(n_1(\texttt{nbits}(z) - 1) + n_3\big)\mathbf{S}_c + (n_4+2)\mathbf{f}. \end{aligned} \tag{7}$$

In the following, we take BW14-351 as an example to analyze the detailed operation count of pairing computation.

Example 1. The selected seed z of BW14-351 has $\texttt{nbits}(z) = 23$ and $\texttt{hw}(z) = 4$. Then, it follows from Eq. (6) that the cost of the Miller Loop is:

$$\begin{aligned} ML = &22(\mathbf{M} + \mathbf{S} + 5\tilde{\mathbf{m}} + \tilde{\mathbf{m}}_u + \tilde{\mathbf{s}}_u + \tilde{\mathbf{m}}_\xi + 3\tilde{\mathbf{s}} + 21\mathbf{m} + \tilde{\mathbf{r}} + 13\tilde{\mathbf{a}} + \mathbf{a}) + \\ &3(\mathbf{M} + 6\tilde{\mathbf{m}} + 4\tilde{\mathbf{m}}_u + \tilde{\mathbf{m}}_\xi + 3\tilde{\mathbf{s}} + 21\mathbf{m} + 2\tilde{\mathbf{r}} + 12\tilde{\mathbf{a}}) + (28\mathbf{M} + 22\mathbf{S} + \mathbf{f}) \\ = &53\mathbf{M} + 44\mathbf{S} + 128\tilde{\mathbf{m}} + 34\tilde{\mathbf{m}}_u + 75\tilde{\mathbf{s}} + 22\tilde{\mathbf{s}}_u + 25\tilde{\mathbf{m}}_\xi + 525\mathbf{m} + 28\tilde{\mathbf{r}} + \mathbf{f} + 322\tilde{\mathbf{a}} + 22\mathbf{a} \\ = &216\tilde{\mathbf{m}} + 193\tilde{\mathbf{m}}_u + 219\tilde{\mathbf{m}}_\xi + 75\tilde{\mathbf{s}} + 22\tilde{\mathbf{s}}_u + 134\tilde{\mathbf{r}} + \mathbf{f} + 525\mathbf{m} + 966\tilde{\mathbf{a}} + 22\mathbf{a} \\ = &537\mathbf{m} + 11271\mathbf{m}_u + 582\mathbf{s}_u + 83543\mathbf{a} + 2975\mathbf{r} \\ = &11808\mathbf{m}_u + 582\mathbf{s}_u + 83543\mathbf{a} + 3512\mathbf{r}. \end{aligned}$$

Table 7. Costs of arithmetic operations in a tower extension field $\mathbb{F}_{p^k}$ on the five candidate curves.

curve	$\tilde{\mathbf{m}} = \tilde{\mathbf{m}}_u + \tilde{\mathbf{r}}$	$\tilde{\mathbf{s}} = \tilde{\mathbf{s}}_u + \tilde{\mathbf{r}}$	$\tilde{\mathbf{i}}$	$\tilde{\mathbf{m}}_\xi$, $\tilde{\mathbf{m}}_{a'}$
BW10-480	$15\mathbf{m}_u+122\mathbf{a}+5\mathbf{r}$	$7\mathbf{m}_u+6\mathbf{s}_u+84\mathbf{a}+5\mathbf{r}$	$\approx \mathbf{i}+2\tilde{\mathbf{m}}+22\mathbf{m}$	$5\mathbf{a}$, $10\mathbf{a}$
BW10-511	$15\mathbf{m}_u+98\mathbf{a}+5\mathbf{r}$	$7\mathbf{m}_u+6\mathbf{s}_u+60\mathbf{a}+5\mathbf{r}$	$\approx \mathbf{i}+2\tilde{\mathbf{m}}+22\mathbf{m}$	$2\mathbf{a}$, -
BW10-512	$15\mathbf{m}_u+122\mathbf{a}+5\mathbf{r}$	$7\mathbf{m}_u+8\mathbf{s}_u+84\mathbf{a}+5\mathbf{r}$	$\approx \mathbf{i}+2\tilde{\mathbf{m}}+22\mathbf{m}$	$5\mathbf{a}$, $10\mathbf{a}$
BW14-351	$24\mathbf{m}_u+162\mathbf{a}+7\mathbf{r}$	$15\mathbf{m}_u+6\mathbf{s}_u+106\mathbf{a}+7\mathbf{r}$	$\approx \mathbf{i}+3\tilde{\mathbf{m}}+38\mathbf{m}$	$\mathbf{a}$, -
BW14-382	$24\mathbf{m}_u+210\mathbf{a}+7\mathbf{r}$	$15\mathbf{m}_u+6\mathbf{s}_u+154\mathbf{a}+7\mathbf{r}$	$\approx \mathbf{i}+3\tilde{\mathbf{m}}+38\mathbf{m}$	$5\mathbf{a}$, $10\mathbf{a}$
$\mathbf{S}_c$	$\mathbf{S}$	$\mathbf{M}$	$\mathbf{I}$	$\mathbf{f}$
$\tilde{\mathbf{m}}+\tilde{\mathbf{s}}+2\tilde{\mathbf{a}}$	$2\tilde{\mathbf{m}}+5\tilde{\mathbf{a}}+2\tilde{\mathbf{m}}_\xi$	$3\tilde{\mathbf{m}}_u+8\tilde{\mathbf{a}}+2\tilde{\mathbf{m}}_\xi+2\tilde{\mathbf{r}}$	$\tilde{\mathbf{i}}+2\tilde{\mathbf{m}}+\tilde{\mathbf{m}}_\xi+2\tilde{\mathbf{s}}+\tilde{\mathbf{a}}$	$(k-2)\mathbf{m}$

Furthermore, it can be obtained from Table 5 that the parameters n_1, n_2, n_3 and n_4 are equal to 14, 12, 0 and 6, respectively. By Eq. (7), the cost of the final exponentiation is:

$$\begin{aligned} FE &= (\mathbf{I}+3\mathbf{M}+2\mathbf{f})+(14\mathbf{e}+12\mathbf{M}+6\mathbf{f}) = \mathbf{I}+57\mathbf{M}+308\mathbf{S}_c+8\mathbf{f} \\ &= \tilde{\mathbf{i}}+310\tilde{\mathbf{m}}+171\tilde{\mathbf{m}}_u+115\tilde{\mathbf{m}}_\xi+310\tilde{\mathbf{s}}+114\tilde{\mathbf{r}}+96\mathbf{m}+1073\tilde{\mathbf{a}} \\ &= \mathbf{i}+16266\mathbf{m}_u+134\mathbf{m}+1860\mathbf{s}_u+5159\mathbf{r}+118894\mathbf{a} \\ &= \mathbf{i}+16400\mathbf{m}_u+1860\mathbf{s}_u+118894\mathbf{a}+5293\mathbf{r}. \end{aligned}$$

In total, the cost of pairing computation on BW14-351 is

$$ML + \text{FE} = \mathbf{i}+28208\mathbf{m}_u+2442\mathbf{s}_u+202437\mathbf{a}+8805\mathbf{r}.$$

Table 8. Operation Counts of pairing computation on the five candidate pairing-friendly curves.

curve	ML	FE	ML+FE
BW10-480	$12861\mathbf{m}_u+1290\mathbf{s}_u$ $+115357\mathbf{a}+4761\mathbf{r}$	$\mathbf{i}+11591\mathbf{m}_u+2412\mathbf{s}_u$ $+111610\mathbf{a}+4682\mathbf{r}$	$\mathbf{i}+24452\mathbf{m}_u+3702\mathbf{s}_u$ $+226967\mathbf{a}+9443\mathbf{r}$
BW10-511	$10027\mathbf{m}_u+822\mathbf{s}_u$ $+71412\mathbf{a}+3508\mathbf{r}$	$\mathbf{i}+12452\mathbf{m}_u+2706\mathbf{s}_u$ $+94203\mathbf{a}+5130\mathbf{r}$	$\mathbf{i}+22479\mathbf{m}_u+3528\mathbf{s}_u$ $+165615\mathbf{a}+8638\mathbf{r}$
BW10-512	$11761\mathbf{m}_u+1170\mathbf{s}_u$ $+105417\mathbf{a}+4341\mathbf{r}$	$\mathbf{i}+12820\mathbf{m}_u+2610\mathbf{s}_u$ $+123314\mathbf{a}+5130\mathbf{r}$	$\mathbf{i}+24581\mathbf{m}_u+3780\mathbf{s}_u$ $+228731\mathbf{a}+9471\mathbf{r}$
BW14-351	$11808\mathbf{m}_u+582\mathbf{s}_u$ $+83543\mathbf{a}+3512\mathbf{r}$	$\mathbf{i}+16400\mathbf{m}_u+1860\mathbf{s}_u$ $+118894\mathbf{a}+5293\mathbf{r}$	$\mathbf{i}+28208\mathbf{m}_u+2442\mathbf{s}_u$ $+202437a\mathbf{a}+8805\mathbf{r}$
BW14-382	$12594\mathbf{m}_u+720\mathbf{s}_u$ $+115874\mathbf{a}+3874\mathbf{r}$	$\mathbf{i}+19883\mathbf{m}_u+2034\mathbf{s}_u$ $+191396\mathbf{a}+6137\mathbf{r}$	$\mathbf{i}+32477\mathbf{m}_u+2754\mathbf{s}_u$ $+307270\mathbf{a}+10011\mathbf{r}$

In Table 8, we summarize the costs of pairing computation on the five candidate curves. It should be noted that the selected primes p for BW10-480, BW10-511, and BW10-512 can be represented by 8 computer words in a 64-bit processor, while for BW14-351 and BW14-382 only require 6 computer words. As illustrated in [4, Sect 8], it is reasonable to estimate that $\mathbf{m}_8 \approx (136/78)\mathbf{m}_6 \approx 1.74\mathbf{m}_6$ and $\mathbf{a}_8 \approx (8/6)\mathbf{a}_6 \approx 1.33\mathbf{a}_6$, where $\mathbf{m}_i$ and $\mathbf{a}_i$ denote the costs of multiplication and addition in $\mathbb{F}_p$, with p a i computer word size prime in a 64-bit processor. Based on the estimate and Table 8, we predict that BW14-351 is the most efficient choice among the five candidate curves for pairing computation.

5 Subgroup Membership Testing

In pairing-based cryptographic protocols, subgroup membership testing plays a critical role in defending against small subgroup attacks. [10,41]. Recent research [17,47] has demonstrated that efficiently computable endomorphisms are powerful tools for accelerating these testings in various pairing groups. In this section, we describe the application of state-of-the-art technique [17] to our specific pairing-friendly curves. Furthermore, we also introduce a faster method for $\mathbb{G}_2$ membership testing.

5.1 $\mathbb{G}_1$ Membership Testing

Given a candidate point P, the process of verifying whether $P \in \mathbb{G}_1$ can be divided into two phases. Concretely, one can first check whether $P \in E(\mathbb{F}_p)$, followed by verifying that the order of P is exactly r. It is clear that the computational cost largely comes from the second phase. Let the endomorphism τ on $\mathbb{G}_1$ act as scalar multiplication by λ_1, and let $\mathcal{L}_\tau$ be a two dimensional lattice as

$$\mathcal{L}_\tau = \{(a_0, a_1) \in \mathbb{Z}^2 | a_0 + a_1 \cdot \lambda_1 \equiv 0 \bmod r\}.$$

By [49, Theorem 2], the norm of the shortest vector in $\mathcal{L}_\tau$ is about $\log r/2$. We let (a_0, a_1) be a vector in $\mathcal{L}_\tau$ with $\gcd(h_1, h_1') = 1$, where $h_1 = \#E(\mathbb{F}_p)/r$ and

$$h_1' = \begin{cases} (a_0^2 - a_0 \cdot a_1 + a_1^2)/r, & \text{if } j(E) = 0; \\ (a_0^2 + a_1^2)/r, & \text{if } j(E) = 1728. \end{cases} \tag{8}$$

Dai et al. [17] prove that the short vector (a_0, a_1) can be used to accelerate $\mathbb{G}_1$ membership testing, i.e.,

$$P \in \mathbb{G}_1 \Leftrightarrow P \in E(\mathbb{F}_p) \text{ and } [a_0]P + [a_1]\tau(P) = \mathcal{O}_E.$$

In general, the constraint $\gcd(h_1, h_1') = 1$ is mild and thus one can find a valid short vector close to the shortest one on many pairing-friendly curves. It means that the process of $\mathbb{G}_1$ membership testing requires about $\log r/2$ iterations.

5.2 $\mathbb{G}_T$ Membership Testing

In the case of $\mathbb{G}_T$ membership testing, the Frobenius endomorphism is critical in finding valid short vectors. To illustrate it, we first use $\mathcal{L}_\pi$ to denote the following $\varphi(k)$ dimensional lattice:

$$\mathcal{L}_\pi = \{(a_0, \cdots, a_{\varphi(k)-1}) \in \mathbb{Z}^{\varphi(k)} | a_0 + a_1 \cdot p + \cdots + a_{\varphi(k)-1} \cdot p^{\varphi(k)-1} \equiv 0 \bmod r\}.$$

The norm of the shortest vector in $\mathcal{L}_\pi$ is about $\log r/\varphi(k)$. For a given short vector $\mathbf{c} = (c_0, c_1, \cdots, c_{\varphi(k)-1}) \in \mathcal{L}_\pi$, we define that

$$h_T = \Phi_k(p)/r \text{ and } h'_T = \sum_{i=0}^{\varphi(k)-1} c_i \cdot p^i.$$

Dai et al. found that if the short vector $\mathbf{c}$ satisfies $\gcd(h_T, h'_T) = 1$, then

$$\alpha \in \mathbb{G}_T \Leftrightarrow \alpha^{\Phi_k(p)} = 1 \text{ and } \prod_{i=0}^{\varphi(k)-1} \alpha^{c_i \cdot p^i} = 1.$$

Likewise, the condition $\gcd(h_T, h'_T) = 1$ is mild, and thus the process of $\mathbb{G}_T$ membership testing requires about $\log r/\varphi(k)$ iterations.

Modified Short Vector: The previous idea for optimizing the final exponentiation still applies to $\mathbb{G}_T$ membership testing such that several full extension field multiplications can be saved. Specifically, once the candidate element α proved to be a member of $\mathbb{G}_{\Phi_k(p)}$, one can replace the original valid vector $\mathbf{c}$ by $\mathbf{c}' = (c_0 + \delta, c_1 - \delta, \cdots, c_{\varphi(k)-1} - \delta, \delta)$ for some integer δ for our target curves as

$$\prod_{i=0}^{\varphi(k)-1} \alpha^{c_i \cdot p^i} = 1 \Leftrightarrow \alpha^{\delta \cdot \Phi_k(p)} \cdot \prod_{i=0}^{\varphi(k)-1} \alpha^{c_i \cdot p^i} = 1.$$

In particular, if the first i tuples of $\mathbf{c}'$ are 0, we then can obtain a new vector as $(c_{i+1} + (-1)^{i+1}\delta, \cdots, c_{\varphi(k)-1} - \delta, \delta, 0, \cdots, 0)$. For instance, using the Magma code provided in [17, **Sect.** 5], a valid vector for $\mathbb{G}_T$ membership testing on BW14-351 is given by $\mathbf{c} = (1, -1, 1, z^2 - 1, -z^2 + z + 1, -z)$. Taking $\delta = -1$, we have

$$(c_0 - 1, c_1 + 1, \cdots, c_6 - 1, 1) = (0, 0, 0, z^2, -z^2 + z, -z + 1, -1).$$

By left-shifting the above vector, a modified short vector $(z^2, -z^2 + z, -z + 1, -1, 0, 0, 0)$ is obtained. Consequently, it is equivalent to checking that

$$\alpha \cdot \alpha^{(p+p^3+p^5)\cdot p} = \alpha^{p+p^3+p^5}, \alpha^{p^3} = \alpha^{z^2} \cdot \alpha^{(z-z^2)\cdot p} \cdot \alpha^{(1-z)\cdot p^2}.$$

5.3 $\mathbb{G}_2$ Membership Testing

Recall from Sect 2.1 that ψ and η represent two efficiently computable endomorphisms on E' with $j(E') = 0$ or 1728. For a given short vector $\mathbf{c} = (c_0, c_1, \cdots, c_{\varphi(k)-1}) \in \mathcal{L}_\pi$, we define that

$$h_2 = \#E'(\mathbb{F}_{p^{k/2}})/r \text{ and } h'_2 = \sum_{i=0}^{\varphi(k)-1} c_i \cdot p^i.$$

Dai et al. method for $\mathbb{G}_2$ membership testing is summarized as follows: If the short vector $\mathbf{c}$ satisfies that $\gcd(h_2, h_2') = 1$, then

$$Q \in \mathbb{G}_2 \Leftrightarrow Q \in E'(\mathbb{F}_{p^{k/2}}) \text{ and } \sum_{i=0}^{\varphi(k)-1} [c_i]\psi^i(Q) = \mathcal{O}_{E'}.$$

Again, the above computation requires about $\log r/\varphi(k)$ iterations. In the following, we develop a faster method for $\mathbb{G}_2$ membership testing, which is tailored to our target curves. To this aim, we first determine the characteristic equation of the endomorphism $\Phi = \psi \circ \eta$.

Lemma 2. *Let E be an ordinary curve over $\mathbb{F}_p$ with $\#E(\mathbb{F}_p) = p + 1 - t$, admitting a twist E'. If $j(E') = 0$ or 1728, then the characteristic equation of Φ is given as follows:*

(1) $j(E') = 0 : \quad \Phi^2 + \frac{t \pm 3f}{2}\Phi + p = 0$ with $t^2 - 4p = -3f^2$;
(2) $j(E') = 1728 : \quad \Phi^2 \pm f\Phi + p = 0$ with $t^2 - 4p = -f^2$.

Proof. We only give the proof of the case $j(E') = 0$ (The proof of the remaining case is similar). As mentioned in Sect 2.1, the characteristic equation of Φ can be expressed as

$$\Phi^2 + m\Phi + n = 0 \tag{9}$$

for some integers m and n. Since $\mathrm{Nrd}(\psi) = p$ and $\mathrm{Nrd}(\eta) = 1$, we have

$$n = \mathrm{Nrd}(\Phi) = \mathrm{Nrd}(\psi) \cdot \mathrm{Nrd}(\eta) = p.$$

Furthermore, the characteristic equation of ψ and η are given as follows:

$$\psi^2 - t\psi + p = 0, \quad \eta^2 + \eta + 1 = 0.$$

It is easy to deduce that

$$\psi = \frac{t \pm \sqrt{-3} \cdot f}{2} \text{ and } \eta = \frac{-1 \pm \sqrt{-3}}{2}.$$

By the fact that $\Phi = \psi \circ \eta$, we have

$$\Phi = \frac{t \pm \sqrt{-3} \cdot f}{2} \cdot \frac{-1 \pm \sqrt{-3}}{2} = \frac{-(t \pm 3f) \pm \sqrt{-3} \cdot (t - f)}{4}. \tag{10}$$

On the other hand, it can be obtained from Eq. (9) that

$$\Phi = \frac{-m \pm \sqrt{m^2 - 4n}}{2}. \tag{11}$$

By comparing Eqs.(10) and (11), we conclude that $m = (t \pm 3f)/2$, which completes the proof. □

Recall that the endomorphism η acts on $\mathbb{G}_2$ as scalar multiplication by λ_2 that is defined in Eq. (3). By combining the actions of ψ and η on $\mathbb{G}_2$ together, we have $\Phi(Q) = [\ell]Q$ for any $Q \in \mathbb{G}_2$, where $\ell = p \cdot \lambda_2 \bmod r$. Since the order of Φ restricting on the $\mathbb{F}_{p^{k/2}}$ rational endomorphism ring is equal to $2k$ or $3k$ on our target curves, we have $r \mid \Phi_{2k}(\ell)$ or $r \mid \Phi_{3k}(\ell)$. The degree of each of the two cyclotomic polynomials is equal to $2\varphi(k)$. For this reason, we can construct the following $2\varphi(k)$ dimensional lattice:

$$\mathcal{L}_\Phi = \{(a_0, \cdots, a_{2\varphi(k)-1}) \in \mathbb{Z}^{2\varphi(k)} | a_0 + a_1 \cdot \ell + \cdots + a_{2\varphi(k)-1} \cdot \ell^{2\varphi(k)-1} \equiv 0 \bmod r\}.$$

Given a short vector $\mathbf{c} = (c_0, c_1, \cdots, c_{2\varphi(k)-1}) \in \mathcal{L}_\Phi$, we define that

$$g(\Phi) = \Phi^2 - t_\Phi \Phi + p \text{ and } h(\Phi) = \sum\nolimits_{i=0}^{2\varphi(k)-1} c_i \Phi^i,$$

where t_Φ is the trace of Φ that is given in Lemma 2. By taking full advantage of the endomorphism Φ, a new method for $\mathbb{G}_2$ membership testing is proposed, which is tailored to our target curves.

Theorem 1. *Let E be an ordinary curve over $\mathbb{F}_p$ with j-invariant 0 or 1728. Let r be a large prime such that $r \mid \#E(\mathbb{F}_p)$. Suppose E admit a twist E' of degree 2 such that $r \mid \#E'(\mathbb{F}_{p^{k/2}})$. Let $\mathbf{c} = (c_0, c_1, \cdots, c_{2\varphi(k)-1}) \in \mathcal{L}_\Phi$, and let $\mathbf{Res}(h(\Phi), g(\Phi))$ be the resultant of $h(\Phi)$ and $g(\Phi)$. Assume that the short vector $\boldsymbol{c}$ satisfies that*

$$\gcd(\boldsymbol{Res}\big(h(\Phi), g(\Phi)), h_2 \cdot r\big) = r. \tag{12}$$

For any non-identity point Q of $E'(\mathbb{F}_{p^{k/2}})$, the point $Q \in \mathbb{G}_2 = E'(\mathbb{F}_{p^{k/2}})[r]$ if and only if

$$\sum\nolimits_{i=0}^{2\varphi(k)-1} [c_i]\Phi^i(Q) = \mathcal{O}_{E'}. \tag{13}$$

Proof. If $Q \in \mathbb{G}_2$, then we have $\Phi(Q) = [\ell]Q$. As a result, we can easily check that

$$\sum\nolimits_{i=0}^{2\varphi(k)-1} [c_i]\Phi^i(Q) = \sum\nolimits_{i=0}^{2\varphi(k)-1} [c_i \ell^i]Q = \mathcal{O}_{E'}.$$

Conversely, we let b_0 and b_1 be two integers satisfying that $b_0 + b_1\Phi = h(\Phi) \bmod g(\Phi)$. By the property of resultant, we have

$$\mathbf{Res}(h(\Phi), g(\Phi)) = \mathbf{Res}(b_0 + b_1\Phi, g(\Phi)) = b_0^2 + b_0 b_1 t_\Phi + b_1^2 p.$$

Furthermore, by the fact that $h(\Phi)(Q) = g(\Phi)(Q) = \mathcal{O}_{E'}$, we have

$$[b_0^2 + b_0 b_1 t_\Phi + b_1^2 p]Q = (b_0 + b_1\hat{\Phi})(b_0 + b_1\Phi)(Q) = \mathcal{O}_{E'}.$$

Therefore, the order of Q divides $\gcd(\mathbf{Res}(h(\Phi), g(\Phi)), h_2 \cdot r)$. Since the selected vector $\mathbf{c}$ is restricted by Eq. (12), we conclude that $Q \in E'(\mathbb{F}_{p^{k/2}})[r] = \mathbb{G}_2$, which completes the proof. $\square$

Likewise, the new approach requires about $\log r/(2\varphi(k))$ bit operations, which is about $2\times$ as fast as the previous leading work [17]. In Table 9, we list the short vectors that can be used for $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ membership testings on the five candidate pairing-friendly curves. It is straightforward to see that the computational cost of $\mathbb{G}_2$ membership testing on the five candidate curves comes largely from a scalar multiplication by z.

Table 9. Short vectors for subgroup membership testings on five candidate pairing-friendly curves.

curve	$\mathbb{G}_1$ (a_0, a_1)	$\mathbb{G}_2$	$\mathbb{G}_T$
BW10-480	$(z^3 - z, -1 - a_0 \cdot z)$	$(1, 0, 0, -z, 0, 0, 0, 0)$	$(z^2, 0, 0, 0, 1)$
BW10-511	$(a_1 \cdot z - 1, z^3 + z^2 - 1)$	$(1, 0, -z - 1, -1, 0, 0, 1, 1)$	$(1, -z^2, 0, z, 0)$
BW10-512	$(z^3 - z, -a_0 \cdot z - 1)$	$(0, 1, 0, z - 1, 0, 1, -z + 1, -1)$	$(1, z^2 - 1, 0, z^2 - 1)$
BW14-351	$(z^5 + z^4 - z^2 - z, (1 - z) \cdot a_0 - 1)$	$(1, 1, 0, -1, -1, 0, 1, 0, -1, -1, 0, z + 1)$	$(z^2, z - z^2, 1 - z, -1)$
BW14-382	$(z^5 - z^3 + z, -1 + a_0 \cdot z)$	$(0, 1, z, -1, 0, 1, 0, -1, 1, 1, 0, z - 1)$	$(z^2, -1, z^2, -1)$

6 Cofactor Multiplication

Hashing a string into $\mathbb{G}_1$ or $\mathbb{G}_2$ is an important building block in pairing-based cryptographic protocols. This operation consists of two phases: first mapping a string into a curve point, followed by a cofactor multiplication so that the resulting point falls into the target subgroup. In this section, we present efficient algorithms for cofactor multiplication for $\mathbb{G}_1$ and $\mathbb{G}_2$ on our chosen target curves.

6.1 Cofactor Multiplication for $\mathbb{G}_1$

Given a random point $P \in E(\mathbb{F}_p)$, cofactor multiplication for $\mathbb{G}_1$ is to map the point P into $\mathbb{G}_1$. The naive way is to perform the scalar multiplication $[h_1]P$, where the cofactor $h_1 = \#E(\mathbb{F}_p)/r$. EI Housni, Guillevic and Piellard [23] observed that the cofactor h_1 can be replaced by a smaller cofactor $\tilde{h}_1$ on a large class of cyclotomic pairing-friendly curves, where $\tilde{h}_1$ is determined by the group structure of $E(\mathbb{F}_p)$:

$$E(\mathbb{F}_p) \cong \mathbb{Z}_{m_1} \oplus \mathbb{Z}_{\tilde{h}_1 \cdot r} \text{ with } m_1 \mid \tilde{h}_1 \text{ and } m_1 \cdot \tilde{h}_1 = h_1.$$

In particular, if the curve E has j-invariant 0 or 1728, then m_1 is the largest integer such that $m_1^2 \mid \#E(\mathbb{F}_p)$ and $m_1 \mid (p - 1)$. Thus, it is not difficult to determine the value of m_1 on the five candidate curves. In the optimal case, we have $m_1 \approx \tilde{h}_1$ and thus the new method would be twice as fast as the naive one, e.g. for the BW10-480 curve.

Faster Cofactor Multiplication for $\mathbb{G}_1$: The algorithm of EI Housni-Guillevic-Piellard can be further optimized in the case that $m_1 \ll \tilde{h}_1$, such as for the BW10-511, BW10-512, BW14-351 and BW14-382 curves. In fact, a random point $P \in E(\mathbb{F}_p)$ can be mapped into $\mathbb{G}_1$ as follows:

$$E(\mathbb{F}_p) \xrightarrow{m_1} E(\mathbb{F}_p)[n_1 \cdot r] \xrightarrow{a_0+a_1\tau} E(\mathbb{F}_p)[r] = \mathbb{G}_1.$$

In detail, the first step is to map the point P into the cyclic group $E(\mathbb{F}_p)[n_1 \cdot r]$ by performing a scalar multiplication by m_1, where $n_1 = \tilde{h}_1/m_1$; the next step is to clear the cofactor n_1 using the endomorphism $a_0 + a_1 \cdot \tau$, where a_0 and a_1 are integers satisfying $a_0 + a_1 \cdot s_1 \equiv 0 \bmod n_1$ and s_1 denotes the scalar of the endomorphism τ acting on $E(\mathbb{F}_p)[n_1 \cdot r]$. More specifically, the LLL algorithm can be exploited to look for two integers a_0 and a_1 such that $\max\{\log|a_0|, \log|a_1|\} \approx \log n_1/2$. In conclusion, cofactor multiplication for $\mathbb{G}_1$ can always be performed in around $\log m_1 + \log n_1/2 \approx \log h_1/2$ iterations, which does not depend on the group structure of $E(\mathbb{F}_p)$. In Table 10, we summarize the parameters h_1, m_1 and $\tilde{h}_1$, and short vectors (a_0, a_1) across different pairing-friendly curves.

Table 10. Important parameters for cofactor multiplication for $\mathbb{G}_1$ on the five candidate pairing-friendly curves.

curve	h_1	m_1	$\tilde{h}_1$	n_1	(a_0, a_1)
BW10-480	$\frac{z^4}{4}$	$\frac{z^2}{2}$	$\frac{z^2}{2}$	1	$-$
BW10-511	$\frac{(z^2-z+1)(z^3-1)^2}{3}$	$\frac{(z^3-1)}{3}$	$(z^2-z+1)(z^3-1)$	$3(z^2-z+1)$	$(1, z)$
BW10-512	$\frac{(z^2-1)^2(z^2+1)}{4}$	$\frac{(z^2-1)}{2}$	$\frac{(z^2-1)(z^2+1)}{2}$	z^2+1	$(z, -1)$
BW14-351	$\frac{(z^2-z+1)(z^2+z+1)}{3}$	1	$\frac{(z^2-z+1)(z^2+z+1)}{3}$	$\frac{(z^2-z+1)(z^2+z+1)}{3}$	$(2z, z^2+z-1)$
BW14-382	$\frac{(z^2-1)^2(z^2+1)}{4}$	$\frac{(z^2-1)}{2}$	$\frac{(z^2-1)(z^2+1)}{2}$	z^2+1	$(z, 1)$

6.2 Cofactor Multiplication for $\mathbb{G}_2$

Cofactor multiplication for $\mathbb{G}_2$ aims to map a random point Q of $E'(\mathbb{F}_{p^{k/2}})$ into $\mathbb{G}_2$. The naive way is to compute $[h_2]Q$ directly, where $h_2 = \#E'(\mathbb{F}_{p^{k/2}})/r$. Since the cofactor h_2 is much larger than the cofactor h_1 and $\mathbb{G}_2$ is defined over $\mathbb{F}_{p^{k/2}}$, the computational cost of the cofactor multiplication for $\mathbb{G}_2$ is more expensive than that for $\mathbb{G}_1$. To date, the fastest known algorithm [25] requires approximately $\log h_2/\varphi(k)$ iterations to clear the cofactor. Recently, Dai et al. [19] proposed a fast method for this operation on curves with the lack of twists. In this subsection, we show that this method can be generalized to our target curves such that the number of iterations can be further reduced to $\log h_2/(2\varphi(k))$.

Lemma 3. *Let* $G_0' = \{Q \in E'(\mathbb{F}_{p^{k/2}}) | \Phi_k(\psi)(Q) = \mathcal{O}_{E'}\}$*. Then the order of* G_0' *is precisely equal to* $\frac{\#E'(\mathbb{F}_{p^{k/2}}) \cdot \#E(\mathbb{F}_p)}{\#E(\mathbb{F}_{p^2})}$.

Proof. Let $G_0 = \{Q \in E(\mathbb{F}_{p^k}) | \Phi_k(\pi)(Q) = \mathcal{O}_E\}$. It is easy to see that $G_0 \cong G_0'$ and thus $\#G_0 = \#G_0'$. By [19, Proposition 2], we have

$$\#G_0 = \frac{\#E(\mathbb{F}_{p^k}) \cdot \#E(\mathbb{F}_p)}{\#E(\mathbb{F}_{p^{k/2}}) \cdot \#E(\mathbb{F}_{p^2})}. \tag{14}$$

On the other hand, it can be obtained from [34, Theorem 3] that

$$\#E(\mathbb{F}_{p^k}) = \#E(\mathbb{F}_{p^{k/2}}) \cdot \#E'(\mathbb{F}_{p^{k/2}}). \tag{15}$$

Inserting Eq.(14) into Eq.(15), it yields that

$$\#G_0' = \#G_0 = \frac{\#E'(\mathbb{F}_{p^{k/2}}) \cdot \#E(\mathbb{F}_p)}{\#E(\mathbb{F}_{p^2})}, \tag{16}$$

which completes the proof of this lemma. □

Since $\mathbb{G}_2$ is a subgroup of G_0', we define that $G_0' \cong \mathbb{Z}_{m_2} \oplus \mathbb{Z}_{m_2 \cdot n_2 \cdot r}$ for some integers m_2 and n_2. As a consequence, the process of mapping a random point of $E'(\mathbb{F}_{p^{k/2}})$ into $\mathbb{G}_2$ can be divided into the following three steps:

$$E'(\mathbb{F}_{p^{k/2}}) \rightarrow G_0' \rightarrow E'(\mathbb{F}_{p^{k/2}})[n_2 \cdot r] \rightarrow \mathbb{G}_2.$$

Since the integer $k/2$ is prime for our chosen curves, a random point $Q \in E'(\mathbb{F}_{p^{k/2}})$ can be mapped into the group G_0' under the endomorphism $\psi + 1$. It is clear that the computational cost of operations largely comes from the last step. In the following, we show how to map a random point of $E'(\mathbb{F}_{p^{k/2}})[n_2 \cdot r]$ into $\mathbb{G}_2$. To illustrate it, we first introduce the two lemmas.

Lemma 4. *Let t' be the trace of the $p^{k/2}$ power Frobenius endomorphism of E'. Let $f, f' \in \mathbb{Z}$ be such that $t^2 - 4p = -Df^2$ and $t'^2 - 4p^{k/2} = -Df'^2$, where $-D$ is the square-free part of $t^2 - 4p$. Let H be a cyclic subgroup of G_0' with order $n_2 \cdot r$. Then $\psi(P) = [a]Q$ for any $Q \in H$, where $a = \frac{t \pm f(t'-2)}{2f'} \bmod n_2 \cdot r$.*

Proof. The proof is given in [25, Lemma 2]. □

As illustrated in [25], Lemma 4 induces a fast approach for cofactor multiplication for $\mathbb{G}_2$ in $\log n_2/\varphi(k)$ iterations on a large class of pairing-friendly curves.

Lemma 5. *Let H be a cyclic subgroup of G_0' with order $n_2 \cdot r$. Then $\eta(Q) = [b]Q$ for any $Q \in H$, where*

$$b = \begin{cases} \dfrac{-f \pm (2a - t)}{2f} \bmod n_2 \cdot r, & if j(E) = 0, \\ \dfrac{\pm(2a - t)}{f} \bmod n_2 \cdot r, & if j(E) = 1728. \end{cases}$$

Proof. The proof is derived from [19, Lemma 2]. □

In the following, we propose a more efficient approach for cofactor multiplication for $\mathbb{G}_2$ suitable for curves listed in Table 1. Our main idea is summarized in the theorem below.

Theorem 2. *Let E be an ordinary elliptic curve admitting a degree-2 twist E' over an extension field $\mathbb{F}_{p^{k/2}}$, where k is the even embedding degree. Let H be a cyclic subgroup of G_0'. If the curve E satisfies the following two conditions: (i)$j(E) \in \{0, 1728\}$; (ii)$3 \nmid k$ and $4 \nmid k$, then there exists a polynomial*

$$h(x) = h_0 + h_1 x + \cdots + h_{s-1} x^{s-1} \in \mathbb{Z}[x]$$

such that $h(\Phi)(Q) \in \mathbb{G}_2$ for any $Q \in H$, where $s = 2\varphi(k)$ and $|h_i| < |n_2|^{1/s}$ for $i = 0, \cdots, s-1$.

Proof. Since $\Phi = \psi \circ \eta$, it can be deduced from Lemmas 4 and 5 that $\Phi(Q) = [\lambda_2]Q$, where $\lambda_2 = a \cdot b \bmod n_2 \cdot r$. Under the condition that $3 \nmid k$ and $4 \nmid k$, we can deduce that the order of Φ acting on the group G_0' is $2k$ or $3k$, which means that

$$\begin{cases} \Phi_{3k}(\lambda_2) \equiv 0 \bmod n_2 \cdot r, & \text{if} j(E) = 0; \\ \Phi_{2k}(\lambda_2) \equiv 0 \bmod n_2 \cdot r, & \text{if} j(E) = 1728. \end{cases}$$

In both cases, the degree of the cyclotomic polynomial is $2\varphi(k)$. Analogous to [25, Theorem 1], there exists a polynomial

$$h(x) = h_0 + h_1 x + \cdots + h_{\varphi(k)-1} x^{2\varphi(k)-1} \in \mathbb{Z}[x]$$

such that $h(\lambda_2)$ is a multiple of n_2, where $|h_i| < |n|^{1/2\varphi(k)}$. Therefore, we have $h(\Phi)Q \in \mathbb{G}_2$ for any $Q \in H$, which completes the proof of this theorem. □

By Theorem 2, the number of iterations for $\mathbb{G}_2$ cofactor multiplication can be reduced to $\frac{\log n_2}{2\varphi(k)} \approx \frac{\log h_2}{2\varphi(k)}$ on the curves listed in Table 1, which is faster than the previous leading work [25]. In the following, we take the BW14-351 curve as an example to describe the main mechanics of the new algorithm.

Example 1 *(Cofactor multiplication for $\mathbb{G}_2$ on BW14-351).* We first can check that $\gcd(\#G_0', p^7 - 1) = 1$ on BW14-351, where $\#G_0'$ can be obtained from Lemma 3. It follows from [19, Proposition 1] that G_0' is cyclic. Applying the LLL algorithm, we can obtain a target vector $(h_0, h_1, \cdots, h_{11})$, where

$$h_i = \begin{cases} 0, & \text{if } 9 \le i \le 11; \\ 2, & \text{if } i = 8; \\ z^2 + z + 1, & \text{if } i = 6; \\ z h_{i+1}, & \text{if } 2 \le i \le 5; \\ z h_2 - 1, & \text{if } i = 1; \\ h_1 + h_4 - h_3 - h_6 + z + 2, & \text{if } i = 0; \\ -h_1 - h_4 + h_2 + h_5 + 1. & \text{if } i = 7. \end{cases}$$

Given a random point $Q \in E'(\mathbb{F}_{p^7})$, we fist obtain the point $P = (\psi + 1)(Q)$. Then, we have $h(\Phi)P = \sum_{i=0}^{8} \Phi^i(R_i) \in \mathbb{G}_2$, where R_i is given as follows:

$$\begin{aligned}
R_8 &= [2]P,\\
R_6 &= [z^2 + z + 1]P,\\
R_i &= [z]R_{i+1},\ 2 \le i \le 5,\\
R_1 &= [z]R_2 - P,\\
R_7 &= -(R_1 + R_4) + (R_2 + R_5) - P,\\
R_0 &= (R_1 + R_4) - (R_3 + R_6) + [z]P + R_8.
\end{aligned}$$

In total, cofactor multiplication for $\mathbb{G}_2$ on BW14-351 costs seven scalar multiplications by z, twenty one point additions, one ψ map, and eight Φ maps.

7 Implementation Results

We first present Magma code to validate the correctness of our proposed algorithms and formulas. Furthermore, we also provide high-speed software implementation for several important pairing group operations on BW10-511 and BW14-351. These two target curves are the winners for pairing computation among our chosen five candidate curves with embedding degrees 10 and 14, respectively. Our implementation is based on the RELIC toolkit, which is a well-known cryptographic library for building pairing-based cryptographic protocols on popular curves at the updated 128 security level, such as BN446 and BLS12-446. In addition, we have observed that the implementation of pairing group operations on BW13-310 presented in [18] also relies on this library. Therefore, we have integrated our code into RELIC to enable fair performance comparisons between the two target curves and these popular curves. Besides our proposed algorithms, we exploit state-of-the-art techniques to implement the following operations.

- The indifferentiable hashing function $\mathcal{H}_1 : \{0,1\}^* \to \mathbb{G}_1$ can be implemented by using the SwiftEC map [14], followed by a cofactor multiplication by h_1.
- Since the cofactor $h_2 > r$ for our chosen curves, the construction of the indifferentiable hashing function $\mathcal{H}_2 : \{0,1\}^* \to \mathbb{G}_2$ only require the map $\mathbb{F}_{p^{k/2}} \to E'(\mathbb{F}_{p^{k/2}})$ to be well-distributed [39, Sect. 1.2]. Consequently, we can use either the Shallue-van de Woestijne map [48] or the SwiftEC map.
- We employ the GLV method [27] and GLS method [26] to perform group exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_T$, respectively.
- For group exponentiation in $\mathbb{G}_2$ on our target curves, we fortunately find that Dai et al. method [18, Sect. 5] can be exploited to achieve a $2\varphi(k)$-dimensional scalar decomposition.
- In terms of the computation of pairings products, we adopt the strategies proposed [28,46,52] such that the final exponentiation step and the squaring computations at the Lines 3 and 12 of Algorithm 2 can be shared.

Table 11. Benchmarking results of pairing group operations across different pairing-friendly curves reported in 10^3 cycles averaged over 10^4 executions.

Operation\Curve	BLS12-446	BN446	BW13-310	BW10-511	BW14-351
hashing to $\mathbb{G}_1$	327	149	125	621	204
hashing to $\mathbb{G}_2$	1630	1361	16699	11981	7236
exp in $\mathbb{G}_1$	541	791	268	592	362
exp in $\mathbb{G}_2$	918	1394	7247	4621	3531
exp in $\mathbb{G}_T$	1322	2243	1062	1476	1098
test in $\mathbb{G}_1$	389	8	269	723	345
test in $\mathbb{G}_2$	333	487	1176	1262	923
test in $\mathbb{G}_T$	372	540	223	586	384
ML	1554	2480	1719	2819	1600
FE	1835	1589	2579	3872	2337
Single pairing	3389	4069	4298	6691	3937
2-pairings	4439	5717	5640	9016	5205
5-pairings	7614	10532	9621	15621	9008
8-pairings	10790	15349	13603	22191	12811

It should be noted that RELIC supports the GLV decomposition and the SwiftEC map once the associated curve parameters are given. Specifically, fast constant-time evaluation of the SwiftEC map in RELIC is based on the technique proposed in [5].

The implementations are compiled with GCC 11.4.0 and flags `-O3 -funroll- loops -march=native -mtune=native`. The benchmarks are executed on an Intel Core i9-12900K processor running at @3.2 GHz with TurboBoost and hyper-threading features disabled. Table 11 reports detailed performance comparisons for each building block across different curves. The results reveal that BW14-351 outperforms BW10-511 for all pairing group operations. Moreover, BW14-351 exhibits competitive performance compared to mainstream pairing-friendly curves. Specifically, single pairing computation on BW14-351 is slightly faster than that on BN446 and BW13-310, while about 16.2% slower than that on BLS12-446. Regarding group exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_T$, BW14-351 is about 49.4% and 20.4% faster than BLS12-446, 118.5% and 100% faster than BN446, while 35.1% and 3.4% slower than BW13-310. Moreover, compared to BW13-310, BW14-351 benefits from a lighter performance penalty for hashing to $\mathbb{G}_2$ and group exponentiation in $\mathbb{G}_2$, although it remains slower than BN446 and BLS12-446.

These results show that each curve has its own strengths and no one can be said to be perfect. The selection of a curve should be based on a careful analysis of the protocol requirements and a thorough evaluation of the performance trade-offs. The BW14-351 curve may be an appropriate choice if a protocol pursues fast group exponentiations in $\mathbb{G}_1$ and $\mathbb{G}_T$, while wishes to minimize the performance penalty for group exponentiations in $\mathbb{G}_2$. In addition, this curve provides the 149-bit security level on the finite field side, making it advantageous for achieving long-term security.

8 Conclusion

In this paper, we provided a comprehensive research for a list of pairing-friendly curves with embedding degrees 10 and 14. We generalized Dai-Zhang-Zhao algorithm for pairing computation on BW13-310 to our target curves, so that the number of Miller iterations can be reduced to approximately $\log r/(2\varphi(k))$, while the denominator elimination trick still can be applied. We also proposed optimized algorithms for cofactor multiplication for $\mathbb{G}_1$ and $\mathbb{G}_2$, and subgroup membership testing for $\mathbb{G}_2$ on these curves. After checking the correctness of our proposed algorithms via Magma code, we presented high-speed software implementations on the BW10-511 and BW14-351 curves inside the RELIC library, and compared performance tradeoffs with other popular curves at the same security level, including BN446, BLS12-446 and BW13-310. Our results showed that the BW14-351 curve is competitive for building pairing-based cryptographic protocols at the updated 128-bit security level.

Acknowledgment. We would like to thank Dmitrii Koshelev for insightful discussion about hashing to $\mathbb{G}_2$ on pairing-friendly curves. The work was supported by the National Key Research and Development Program of China (No. 2021YFA1000600), the Postdoctoral Fellowship Program of CPSF (N0. GZC20231995), the National Natural Science Foundation of China (Nos. 62325209, 61972428), the Major Program(JD) of Hubei Province (No. 2023BAA027), Guangdong Major Project of Basic and Applied Basic Research (No. 2019B030302008), Guangdong Provincial Key Laboratory of Information Security Technology (No. 2023B1212060026) and the Fundamental Research Funds for the Central Universities (Nos. 2042023KF0203, 2042024kf1013).

References

1. European union agency of network and information security (ENISA): Algorithms, key sizes and parameters report. https://www.enisa.europa.eu/publications/algorithms-key-sizes-and-parameters-report (2013)
2. Aranha, D.F., Gouvêa, C.P.L.: RELIC is an Efficient LIbrary for Cryptography, https://github.com/relic-toolkit/relic
3. Aranha, D.F., El Housni, Y., Guillevic, A.: A survey of elliptic curves for proof systems. Designs, Codes and Cryptography **91**(11), 3333–3378 (Nov 2023). https://doi.org/10.1007/s10623-022-01135-y
4. Aranha, D.F., Fuentes-Castañeda, L., Knapp, E., Menezes, A., Rodríguez-Henríquez, F.: Implementing pairings at the 192-bit security level. In: Abdalla, M., Lange, T. (eds.) Pairing-Based Cryptography – Pairing 2012. pp. 177–195. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36334-4_11
5. Aranha, D.F., Hvass, B.S., Spitters, B., Tibouchi, M.: Faster constant-time evaluation of the kronecker symbol with application to elliptic curve hashing. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. p. 3228-238. CCS '23, Association for Computing Machinery, New York, NY, USA (2023).https://doi.org/10.1145/3576915.3616597

6. Aranha, D.F., Karabina, K., Longa, P., Gebotys, C.H., López, J.: Faster explicit formulas for computing pairings over ordinary curves. In: Paterson, K.G. (ed.) Advances in Cryptology – EUROCRYPT 2011. pp. 48–68. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20465-4_5
7. Azarderakhsh, R., Fishbein, D., Grewal, G., Hu, S., Jao, D., Longa, P., Verma, R.: Fast software implementations of bilinear pairings. IEEE Transactions on Dependable and Secure Computing **14**(6), 605–619 (2017). https://doi.org/10.1109/TDSC.2015.2507120
8. Barbulescu, R., Duquesne, S.: Updating key size estimations for pairings. Journal of Cryptology **32**(4), 1298–1336 (2019). https://doi.org/10.1007/s00145-018-9280-5
9. Barbulescu, R., Gaudry, P., Kleinjung, T.: The tower number field sieve. In: Iwata, T., Cheon, J.H. (eds.) Advances in Cryptology – ASIACRYPT 2015. pp. 31–55. Springer Berlin Heidelberg, Berlin, Heidelberg (2015). https://doi.org/10.1007/978-3-662-48800-3_2
10. Barreto, P.S.L.M., Costello, C., Misoczki, R., Naehrig, M., Pereira, G.C.C.F., Zanon, G.: Subgroup security in pairing-based cryptography. In: Lauter, K., Rodríguez-Henríquez, F. (eds.) Progress in Cryptology – LATINCRYPT 2015. pp. 245–265. Springer International Publishing, Cham (2015). https://doi.org/10.1007/978-3-319-22174-8_14
11. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Preneel, B., Tavares, S. (eds.) Selected Areas in Cryptography – SAC 2005. pp. 319–331. Springer Berlin Heidelberg, Berlin, Heidelberg (2006).https://doi.org/10.1007/11693383_22
12. Brezing, F., Weng, A.: Elliptic curves suitable for pairing based cryptography. Designs, Codes and Cryptography **37**(1), 133–141 (Oct 2005). https://doi.org/10.1007/s10623-004-3808-4
13. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: Proceedings of the 11th ACM Conference on Computer and Communications Security. pp. 132–145. Association for Computing Machinery, New York, NY, USA (2004). https://doi.org/10.1145/1030083.1030103
14. Chavez-Saab, J., Rodríguez-Henríquez, F., Tibouchi, M.: SwiftEC: Shallue-van de woestijne indifferentiable function to elliptic curves: Faster indifferentiable hashing to elliptic curves. In: Advances in Cryptology – ASIACRYPT 2022. pp. 63–92. Springer-Verlag, Berlin, Heidelberg (2023). https://doi.org/10.1007/978-3-031-22963-3_3
15. Clarisse, R., Duquesne, S., Sanders, O.: Curves with fast computations in the first pairing group. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) Cryptology and Network Security – CNS2020. pp. 280–298. Springer International Publishing, Cham (2020).https://doi.org/10.1007/978-3-030-65411-5_14
16. Costello, C., Lange, T., Naehrig, M.: Faster pairing computations on curves with high-degree twists. In: Nguyen, P.Q., Pointcheval, D. (eds.) Public Key Cryptography – PKC 2010. pp. 224–242. Springer Berlin Heidelberg, Berlin, Heidelberg (2010).https://doi.org/10.1007/978-3-642-13013-7_14
17. Dai, Y., Lin, K., Zhao, C.A., Zhou, Z.: Fast subgroup membership testings for $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ on pairing-friendly curves. Designs, Codes and Cryptography **91**(10), 3141–3166 (2023). https://doi.org/10.1007/s10623-023-01223-7
18. Dai, Y., Zhang, F., Zhao, C.A.: Don't forget pairing-friendly curves with odd prime embedding degrees. IACR Transactions on Cryptographic Hardware and Embedded Systems **2023**(4), 393–419 (2023). https://doi.org/10.46586/tches.v2023.i4.393-419

19. Dai, Y., Zhang, F., Zhao, C.A.: Fast hashing to $\mathbb{G}_2$ on pairing-friendly curves with the lack of twists. Finite Fields and Their Applications **91**, 102–263 (2023). https://doi.org/10.1016/j.ffa.2023.102263
20. De Feo, L., Masson, S., Petit, C., Sanso, A.: Verifiable delay functions from supersingular isogenies and pairings. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology – ASIACRYPT 2019. pp. 248–277. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-34578-5_10
21. El Housni, Y., Guillevic, A.: Optimized and secure pairing-friendly elliptic curves suitable for one layer proof composition. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) Cryptology and Network Security – CANS 2020. pp. 259–279. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-65411-5_13
22. El Housni, Y., Guillevic, A.: Families of SNARK-friendly 2-chains of elliptic curves. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology – EUROCRYPT 2022. pp. 367–396. Springer International Publishing, Cham (2022). https://doi.org/10.1007/978-3-031-07085-3_13
23. El Housni, Y., Guillevic, A., Piellard, T.: Co-factor clearing and subgroup membership testing on pairing-friendly curves. In: Batina, L., Daemen, J. (eds.) Progress in Cryptology – AFRICACRYPT 2022. pp. 518–536. Springer Nature Switzerland, Cham (2022).https://doi.org/10.1007/978-3-031-17433-9_22
24. Freeman, D., Scott, M., Teske, E.: A taxonomy of pairing-friendly elliptic curves. Journal of Cryptology **23**(2), 224–280 (2010). https://doi.org/10.1007/s00145-009-9048-z
25. Fuentes-Castañeda, L., Knapp, E., Rodríguez-Henríquez, F.: Faster hashing to $\mathbb{G}_2$. In: Miri, A., Vaudenay, S. (eds.) Selected Areas in Cryptography – SAC 2011. pp. 412–430. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28496-0_25
26. Galbraith, S.D., Lin, X., Scott, M.: Endomorphisms for faster elliptic curve cryptography on a large class of curves. In: Joux, A. (ed.) Advances in Cryptology - EUROCRYPT 2009. pp. 518–535. Springer Berlin Heidelberg, Berlin, Heidelberg (2009)
27. Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster point multiplication on elliptic curves with efficient endomorphisms. In: Kilian, J. (ed.) Advances in Cryptology — CRYPTO 2001. pp. 190–200. Springer Berlin Heidelberg, Berlin, Heidelberg (2001).https://doi.org/10.1007/3-540-44647-8_11
28. Granger, R., Smart, N.P.: On computing products of pairings. Cryptology ePrint Archive, Paper 2006/172 (2006), https://eprint.iacr.org/2006/172
29. Granger, R., Scott, M.: Faster squaring in the cyclotomic subgroup of sixth degree extensions. In: Nguyen, P.Q., Pointcheval, D. (eds.) Public Key Cryptography – PKC 2010. pp. 209–223. Springer Berlin Heidelberg, Berlin, Heidelberg (2010).https://doi.org/10.1007/978-3-642-13013-7_13
30. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016. pp. 305–326. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_11
31. Guillevic, A.: A short-list of pairing-friendly curves resistant to special TNFS at the 128-bit security level. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) Public-Key Cryptography – PKC 2020. pp. 535–564. Springer International Publishing, Cham (2020). https://doi.org/10.1007/978-3-030-45388-6_19

32. Guillevic, A., Masson, S., Thomé, E.: Cocks-Pinch curves of embedding degrees five to eight and optimal ate pairing computation. Designs, Codes and Cryptography **88**(6), 1047–1081 (2020). https://doi.org/10.1007/s10623-020-00727-w
33. Guillevic, A., Singh, S.: On the alpha value of polynomials in the tower number field sieve algorithm. Mathematical Cryptology **1**(1) (Feb 2021), open access at https://journals.flvc.org/mathcryptology/article/view/125142
34. Hess, F., Smart, N.P., Vercauteren, F.: The Eta pairing revisited. IEEE Transactions on Information Theory **52**(10), 4595–4602 (2006). https://doi.org/10.1109/TIT.2006.881709
35. Ionica, S., Robert, D.: Pairings. In: El Mrabet, N., Joye, M. (eds.) Guide to pairing-based cryptography. Chapman and Hall/CRC Press, BocaRaton (2016)
36. Joux, A., Pierrot, C.: The special number field sieve in $\mathbb{F}_{p^n}$. In: Cao, Z., Zhang, F. (eds.) Pairing-Based Cryptography – Pairing 2013. pp. 45–61. Springer International Publishing, Cham (2014). https://doi.org/10.1007/978-3-319-04873-4_3
37. Karatsuba, A.: Multiplication of multidigit numbers on automata. In: Soviet physics doklady. vol. 7, pp. 595–596 (1963)
38. Kim, T., Barbulescu, R.: Extended tower number field sieve: A new complexity for the medium prime case. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016. pp. 543–571. Springer Berlin Heidelberg, Berlin,Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_20
39. Koshelev, D.: Some remarks on how to hash faster onto elliptic curves. Journal of Computer Virology and Hacking Techniques (Mar 2024). https://doi.org/10.1007/s11416-024-00514-4
40. Lidl, R., Niederreiter, H.: Introduction to finite fields and their applications. Cambridge university press (1994)
41. Lim, C.H., Lee, P.J.: A key recovery attack on discrete log-based schemes using a prime order subgroup. In: Kaliski, B.S. (ed.) Advances in Cryptology — CRYPTO 1997. pp. 249–263. Springer Berlin Heidelberg, Berlin, Heidelberg (1997). https://doi.org/10.1007/BFb0052240
42. Miller, V.S.: The Weil pairing, and its efficient calculation. Journal of Cryptology **17**(4), 235–261 (2004). https://doi.org/10.1007/s00145-004-0315-8
43. Pollard, J.M.: A Monte Carlo method for factorization. Bit Numerical Mathematics **15**(3), 331–334 (1975). https://doi.org/10.1007/BF01933667
44. Schirokauer, O.: Discrete logarithms and local units. Philosophical Transactions: Physical Sciences and Engineering **345**(1676), 409–423 (1993). https://doi.org/10.1098/rsta.1993.0139
45. Scott, M.: Implementing cryptographic pairings. In: Proceedings of the First International Conference on Pairing-Based Cryptography – Pairing 2007. p. 177-196. Springer-Verlag, Berlin, Heidelberg (2007)
46. Scott, M.: On the efficient implementation of pairing-based protocols. In: Chen, L. (ed.) Cryptography and Coding. pp. 296–308. Springer Berlin Heidelberg, Berlin, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25516-8_18
47. Scott, M.: A note on group membership tests for $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ on BLS pairing-friendly curves. Cryptology ePrint Archive, Report 2021/1130 (2021), https://ia.cr/2021/1130
48. Shallue, A., van de Woestijne, C.E.: Construction of rational points on elliptic curves over finite fields. In: Hess, F., Pauli, S., Pohst, M. (eds.) Algorithmic Number Theory Symposium– ANTS 2006. pp. 510–524. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
49. Vercauteren, F.: Optimal pairings. IEEE Transactions on Information Theory **56**(1), 455–461 (2009).https://doi.org/10.1109/TIT.2009.2034881

50. Washington, L.C.: Elliptic curves: number theory and cryptography. Chapman and Hall/CRC (2008)
51. Yang, K., Chen, L., Zhang, Z., Newton, C.J., Yang, B., Xi, L.: Direct anonymous attestation with optimal TPM signing efficiency. IEEE Transactions on Information Forensics and Security **16**, 2260–2275 (2021). https://doi.org/10.1109/TIFS.2021.3051801
52. Zhang, X., Lin, D.: Analysis of optimum pairing products at high security levels. In: Galbraith, S., Nandi, M. (eds.) Progress in Cryptology - INDOCRYPT 2012. pp. 412–430. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
53. Zhao, C.A., Zhang, F., Huang, J.: A note on the ate pairing. International Journal of Information Security **7**(6), 379–382 (Nov 2008). https://doi.org/10.1007/s10207-008-0054-1

Author Index