

Kai-Min Chung
Yu Sasaki (Eds.)

LNCS 15488

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 V

5
Part V



 Springer

Lecture Notes in Computer Science

15488


Founding Editors

Gerhard Goos
Juris Hartmanis

Editorial Board Members

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

Wen Gao, *Peking University, Beijing, China*

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

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

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


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


Kai-Min Chung · Yu Sasaki
Editors

Advances in Cryptology – ASIACRYPT 2024

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

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 978-981-96-0934-5

ISBN 978-981-96-0935-2 (eBook)

<https://doi.org/10.1007/978-981-96-0935-2>

© International Association for Cryptologic Research 2025

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

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

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

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

If disposing of this product, please recycle the paper.

Preface

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

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

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

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

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

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

The program of Asiacrypt 2024 also featured the 2024 IACR Distinguished Lecture, delivered by Paul Kocher, as well as an invited talk by Dakshita Khurana. Following Eurocrypt 2024, we selected seven PC members for the Distinguished PC Members Awards, nominated by the area chairs and program chairs. The Distinguished PC Members Awards went to Sherman S. M. Chow, Elizabeth Crites, Matthias J. Kannwischer, Mustafa Khairallah, Ruben Niederhagen, Maciej Obremski and Keita Xagawa.

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

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

December 2024

Kai-Min Chung
Yu Sasaki

Organization

General Chair

Bimal Kumar Roy

TCG CREST Kolkata, India

Program Committee Chairs

Kai-Min Chung

Academia Sinica, Taiwan

Yu Sasaki

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

Area Chairs

Prabhanjan Ananth

University of California, Santa Barbara, USA

Siyao Guo

NYU Shanghai, China

Goichiro Hanaoka

National Institute of Advanced Industrial Science
and Technology, Japan

Tetsu Iwata

Nagoya University, Japan

Arpita Patra

Indian Institute of Science Bangalore, India

Bo-Yin Yang

Academia Sinica, Taiwan

Program Committee

Akshima

NYU Shanghai, China

Bar Alon

Ben-Gurion University, Israel

Elena Andreeva

TU Wien, Austria

Nuttapong Attrapadung

AIST, Japan

Subhadeep Banik

University of Lugano, Switzerland

Zhenzhen Bao

Tsinghua University, China

James Bartusek

University of California, Berkeley, USA

Hanno Becker

Amazon Web Services, UK

Sonia Belaïd

CryptoExperts, France

Ward Beullens

IBM Research, Switzerland

Andrej Bogdanov

University of Ottawa, Canada

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

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

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

Additional Reviewers

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

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

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

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

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

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

Artifact Chair

Rei Ueno

Kyoto University, Japan

Artifact Evaluation Committee

Julien Béguinot	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 V

Key Exchange Protocols

C'est Très CHIC: A Compact Password-Authenticated Key Exchange from Lattice-Based KEM	3
<i>Afonso Arriaga, Manuel Barbosa, Stanislaw Jarecki, and Marjan Škrobot</i>	
Efficient Asymmetric PAKE Compiler from KEM and AE	34
<i>You Lyu, Shengli Liu, and Shuai Han</i>	
Threshold PAKE with Security Against Compromise of All Servers	66
<i>Yanqi Gu, Stanislaw Jarecki, Pawel Kedzior, Phillip Nazarian, and Jiayu Xu</i>	
Key Exchange in the Post-snowden Era: Universally Composable Subversion-Resilient PAKE	101
<i>Suvradip Chakraborty, Lorenzo Magliocco, Bernardo Magri, and Daniele Venturi</i>	
Tightly-Secure Group Key Exchange with Perfect Forward Secrecy	134
<i>Emanuele Di Giandomenico, Doreen Riepel, and Sven Schäge</i>	
Anamorphic Authenticated Key Exchange: Double Key Distribution Under Surveillance	168
<i>Weihao Wang, Shuai Han, and Shengli Liu</i>	

Succinct Arguments

RoK, Paper, SISsors Toolkit for Lattice-Based Succinct Arguments: (Extended Abstract)	203
<i>Michael Kloof, Russell W. F. Lai, Ngoc Khanh Nguyen, and Michal Osadnik</i>	
MuxProofs: Succinct Arguments for Machine Computation from Vector Lookups	236
<i>Zijing Di, Lucas Xia, Wilson Nguyen, and Nirvan Tyagi</i>	

Verifiable Computation

Proofs for Deep Thought: Accumulation for Large Memories and Deterministic Computations	269
<i>Benedikt Bünz and Jessica Chen</i>	

HELIOPOLIS: Verifiable Computation over Homomorphically Encrypted Data from Interactive Oracle Proofs is Practical	302
<i>Diego F. Aranha, Anamaria Costache, Antonio Guimarães, and Eduardo Soria-Vazquez</i>	

Zero-knowledge Protocols

Interactive Line-Point Zero-Knowledge with Sublinear Communication and Linear Computation	337
<i>Fuchun Lin, Chaoping Xing, and Yizhou Yao</i>	

LogRobin++: Optimizing Proofs of Disjunctive Statements in VOLE-Based ZK	367
<i>Carmit Hazay, David Heath, Vladimir Kolesnikov, Muthuramakrishnan Venkatasubramanian, and Yibin Yang</i>	

FLI: Folding Lookup Instances	402
<i>Albert Garreta and Ignacio Manzur</i>	

Code-Based Zero-Knowledge from VOLE-in-the-Head and Their Applications: Simpler, Faster, and Smaller	436
<i>Ying Ouyang, Deng Tang, and Yanhong Xu</i>	

Author Index	471
---------------------------	-----

Key Exchange Protocols



C'est Très CHIC: A Compact Password-Authenticated Key Exchange from Lattice-Based KEM

Afonso Arriaga¹✉ , Manuel Barbosa^{2,3,4} , Stanislaw Jarecki⁵ ,
and Marjan Škrobot¹ 

¹ SnT - University of Luxembourg, Esch-sur-Alzette, Luxembourg
{afonso.deleue,marjan.skrobot}@uni.lu

² FCUP, University of Porto, Porto, Portugal
mbb@fc.up.pt

³ INESC TEC, Porto, Portugal

⁴ Max Planck Institute for Security and Privacy, Bochum, Germany

⁵ University of California, Irvine, USA
stanislawjarecki@gmail.com

Abstract. Driven by the NIST's post-quantum standardization efforts and the selection of Kyber as a lattice-based Key-Encapsulation Mechanism (KEM), several Password Authenticated Key Exchange (PAKE) protocols have been recently proposed that leverage a KEM to create an efficient, easy-to-implement and secure PAKE. In two recent works, Beguinet et al. (ACNS 2023) and Pan and Zeng (ASIACRYPT 2023) proposed generic compilers that transform KEM into PAKE, relying on an Ideal Cipher (IC) defined over a group. However, although IC on a group is often used in cryptographic protocols, special care must be taken to instantiate such objects in practice, especially when a low-entropy key is used. To address this concern, Dos Santos et al. (EUROCRYPT 2023) proposed a relaxation of the IC model under the Universal Composability (UC) framework called Half-Ideal Cipher (HIC). They demonstrate how to construct a UC-secure PAKE protocol, EKE-KEM, from a KEM and a modified 2-round Feistel construction called m2F. Remarkably, the m2F sidesteps the use of an IC over a group, and instead employs an IC defined over a fixed-length bitstring domain, which is easier to instantiate.

In this paper, we introduce a novel PAKE protocol called CHIC that improves the communication and computation efficiency of EKE-KEM, by avoiding the HIC abstraction. Instead, we split the KEM public key in two parts and use the m2F directly, without further randomization. We provide a detailed proof of the security of CHIC and establish precise security requirements for the underlying KEM, including one-wayness and anonymity of ciphertexts, and uniformity of public keys. Our findings extend to general KEM-based EKE-style protocols and show that a passively secure KEM is not sufficient. In this respect, our results align with those of Pan and Zeng (ASIACRYPT 2023), but contradict the analyses of KEM-to-PAKE compilers by Beguinet et al. (ACNS 2023) and Dos Santos et al. (EUROCRYPT 2023).

Finally, we provide an implementation of CHIC, highlighting its minimal overhead compared to the underlying KEM – Kyber. An interesting aspect of the implementation is that we reuse the rejection sampling procedure in Kyber reference code to address the challenge of hashing onto the public key space. As of now, to the best of our knowledge, CHIC stands as the most efficient PAKE protocol from black-box KEM that offers rigorously proven UC security.

Keywords: Password Authenticated Key Exchange · Key Encapsulation Mechanism · Universal Composability · Post-Quantum · Ideal Cipher

1 Introduction

The problem of attaining secure communication online is commonly addressed by employing Authenticated Key Exchange (AKE) protocols that involve high-entropy long-term private keys, often relying on Public Key Infrastructure (PKI). However, in scenarios where humans are involved in the authentication process, secure storage of long-term private keys by users is impractical, and most applications resort to a simpler and cost-effective solution—human-memorizable passwords. In most cases, applications carry out password-based authentication using (variants of) the bare-bones protocol where the user sends a password across the network to be checked with respect to a previously stored record (usually a salted hashed value) of the same password. This protocol, which is chosen due to its usability and ease of deployment, has a number of disadvantages from the security point of view. An obvious shortcoming is that the password is explicitly transferred across the communications channel, and so it requires a previously established secure and one-side-authenticated channel to the server checking the password. This opens the way to a number of well-known attacks, such as impersonating the server via a phishing attack.

Password Authenticated Key Exchange (PAKE) [6, 7, 10] is a cryptographic primitive that can mitigate some of the limitations associated with low-entropy passwords, and bootstrap a shared password into a cryptographically strong session key. Intuitively, PAKE protocols guarantee that the only way to extract a password from a user over the network is to actively perform a password-guessing attack by trying to run the protocol with the user multiple times.

The most efficient PAKE constructions to date, namely the CPACE protocol that has been recently chosen for standardization by the IETF [2], are built as variants of the Diffie-Hellman protocol and they achieve security with essentially no bandwidth overhead and minimal computational overhead—in CPACE this overhead is reduced to hash operations. Indeed, one of the takeaways of the CPACE selection process was that performance is critical for adoption.¹ This is because target applications include resource-constrained devices (e.g., IoT networks) and ad-hoc contexts (e.g., ePassports and file transfers). Therefore, a natural question to ask in the current context of migration to post-quantum

¹ <https://mailarchive.ietf.org/arch/msg/cfrg/usR4me-MKbW4QO0LprDKXu3TOHY>.

secure cryptography is how to construct efficient PAKE protocols that are not Diffie-Hellman based and that, ideally, can leverage the recent results of the NIST post-quantum competition.

KEM-Based PAKE Protocols. In this direction, and very recently, several works [3, 5, 23, 27, 28] proposed black-box constructions of PAKE from a Key-Encapsulation Mechanism (KEM) and an Ideal Cipher (IC) or its variants (see below).² Conceptually, this KEM-based design paradigm sheds new light on the thirty-year-old *Encrypted Key Exchange* (EKE) approach to PAKE by Bellare and Merritt [7]. From a practical point of view, this recent focus on the generic conversion of KEM into PAKE is largely driven by the efforts of the National Institute of Standards and Technology (NIST) to standardize Post-Quantum (PQ) cryptographic schemes, including KEM and digital signatures. In particular, the standardization of the first post-quantum KEM was just concluded [25] and the scheme is based on *Crystals-Kyber*, a module-lattice-based KEM. *Kyber* has undergone extensive scrutiny regarding its security and anonymity properties, as well as secure and efficient implementation, and this body of research can be leveraged when constructing PAKE protocols that use KEM in a black-box way.

A common characteristic of the above KEM-based PAKE proposals is their reliance on Random Oracle (RO) and Ideal Cipher (IC) models.³ Despite the similarities among these proposed protocols, they still differ in subtle ways and can be categorized based on the model of analysis, design structure, and KEM security properties used to establish PAKE security. The protocols put forth by Bradley et al. [11], McQuoid et al. [23], Beguinet et al. [5] and Dos Santos et al. [28] are analysed under Universal Composability (UC) PAKE framework [13], while Pan and Zeng [27] and Alnahawi et al. [3] prove security under the game-based PAKE definition of Bellare-Pointcheval-Rogaway (BPR) [6]. We note that the UC PAKE security model of Canetti et al. [13] is significantly stronger than the BPR model. The superiority of the former springs fundamentally from the UC framework's ability to capture security under arbitrary correlations of password inputs, which is beyond the scope of current game-based PAKE security notions. Indeed, another important takeaway from the CPACE selection process within the IETF, was the relevance of a (thoroughly scrutinized) proof of security in the UC framework.⁴

Two Approaches to KEM-Based PAKE. Prior KEM-based PAKE protocols follow two distinct design patterns. Firstly, sPAKE [23], CAKE [5], and PAKE-KEM [27], follow a procedure where the initiator Alice employs an IC to encrypt a KEM public key under her password, and the responder Bob decrypts

² This list can be extended by the PAPKE protocol of [11], which was originally presented as a generic PAKE from PKE and IC, but it can be recast as construction from KEM and IC.

³ In [23] security is claimed based solely on RO, but that claim has not been formally established.

⁴ <https://mailarchive.ietf.org/arch/msg/cfrg/47pnOSSrVS8uozXbAuM-alEk0-s>.

this public key and uses it to encapsulate a secret value.⁵ This secret value is used by both parties as an input to a hash function—modeled as a Random Oracle (RO)—to derive a session key. However, Bob does not send the KEM ciphertext to Alice in the clear but instead utilizes a second IC to encrypt the KEM ciphertext before transmitting it to the initiator. This approach ensures that both parties are committed to a single password via IC encryption, based on the collision-freeness of IC outputs. A practical disadvantage of this two-sided usage of IC is that it requires two distinct IC instances, one over the domain of KEM public keys, and the other over the domain of KEM ciphertexts. In lattice-based KEMs, these domains are typically different, and both of them are large, which makes implementing IC for these domains non-trivial.

The second design pattern, employed in protocols PAPKE [11], OCAKE [5], EKE-KEM [28], and PAKEM [3], takes a slightly different approach. Here, the KEM ciphertext obtained by Bob is sent in the clear, accompanied by a key confirmation *tag*, whose purpose is to make Bob’s message a commitment to a single password guess.⁶ The second design uses only one instance of IC, which makes it more efficient, and it does not require special properties of KEM ciphertexts, e.g. that they are indistinguishable from random elements of the ciphertext domain. In this work, we focus on efficiency and therefore adopt this design pattern.

Opening up the IC Blackbox. The sPAKE and EKE-KEM protocols of resp. [23] and [28] deviate from the above pattern by replacing the Ideal Cipher on the domain of public keys (and ciphertexts in [23]) with a weaker and easier-to-construct primitive. One motivation for reducing the requirement on the password-based encryption component is the difficulty of efficiently instantiating IC on a group domain—cf. the discussion of the costs of possible approaches in e.g. [28], which is necessary to instantiate the “KEM+IC” design for PAKE using KEM instantiated as an Elliptic-Curve Diffie-Hellman. However, instantiating the same KEM+IC approach using a lattice-based KEM is also non-trivial because it would require a special-purpose IC on a domain of large bit strings (around one kilobyte in the case of Kyber). Even though there exist methods for extending an IC domain to bitstrings of arbitrary size, e.g., using Feistel networks, [14,17] these generic IC domain extension techniques would add significant complexity to an implementation and incur a significant performance penalty.

Motivated by the above, McQuoid et al. [23] proposed to replace IC in this KEM+IC approach to PAKE with a weaker primitive of a *Programmable-Once Public Function* (POPF), which they showed can be instantiated with a 2-round

⁵ In sPAKE [23] the IC is replaced by a weaker primitive, see more below.

⁶ A seeming exception is the PAPKE protocol [11], which does not attach such a tag explicitly, but it requires a *strong robustness* property of the KEM, and the generic method for achieving this property includes expanding a CCA-secure KEM ciphertext with a key-committing tag [1]. Protocol PAKEM [3] also diverges from the pattern because it employs an additional message flow where Alice sends her own key confirmation tag to Bob. This last message achieves explicit mutual authentication in the Alice-to-Bob direction, but it adds an extra round to the protocol.

Feistel network (2F). In particular, in the case of Kyber KEM, the 2F encryption would involve just one RO hash onto the KEM public key domain, and one RO hash onto a domain of bitstrings of length 3λ , where λ is the security parameter. However, this way of implementing password encryption would add at least 384 ($=3 \times 128$) bits to the KEM ciphertext. Moreover, as mentioned in footnote (see Footnote 3), the analysis of the resulting protocol as a UC-secure PAKE is currently incomplete. Dos Santos et al. [28] modify the 2-round Feistel network used by [23]—calling the result a *modified 2-Feistel* (m2F)—by reducing the bandwidth overhead to 256 ($=2 \times 128$) bits: this is achieved at the cost of adding an IC on 256-bit strings into the encryption procedure. The security proof in [28] shows that m2F realizes a UC abstraction of a (randomized) *Half-Ideal Cipher* (HIC), and then shows that the above KEM+IC approach to UC PAKE works also in the case of KEM+HIC. However, because it is a randomized encryption using a 256-bit random seed, it adds at least 256 bits to the encrypted public key.

Main Contribution: Compact m2F and Bandwidth-Minimal KEM-to-PAKE Compiler. In this paper we revisit the construction of [28] and reduce the bandwidth overhead to a minimum. We observe that, for Kyber and other post-quantum KEMs, the public key can be split into two components, one of which is a 32-byte uniform seed ρ , and ask the following natural question:

Can we reduce the bandwidth overhead of the m2F by using ρ as the ephemeral randomness r in the m2F construction?

We answer this question in the affirmative by giving direct proof that the resulting construction is a UC secure PAKE in the joint Ideal Cipher and Random Oracle model. By *direct proof* we mean that we do not rely on the Half-Ideal-Cipher abstraction of [28], and instead perform the proof over the fully expanded construction. The reason for this is that the notion of a UC-secure Half-Ideal-Cipher crucially relies on the fact that the m2F construction is randomized, i.e., that honest parties choose an ephemeral randomness that is *independent* of the input public key. By unifying this ephemeral randomness with a public-key component we lose this property and the ability to modularize the m2F construction. We call our construction CHIC for Compact Half-Ideal-Cipher, as a way to acknowledge the inspiration in the work of [28].

Second Contribution: Requirements for the KEM. We provide a detailed proof of the security of CHIC and establish precise security requirements for the underlying KEM, including passive one-way security (OW-CPA) and pseudo-uniformity of public-keys (UNI-PK), necessary to achieve UC PAKE security. Like prior works, our proof shows that anonymity is also a necessary property for the security of the construction. However, we show that passive anonymity (i.e., indistinguishability of public keys and ciphertexts) is *not* sufficient to conclude the proof. We show that CHIC requires a ANO-1PCA-secure KEM⁷, and that our analysis extends to the proofs by Beguin et al. [5] and Dos Santos et al. [28], despite the claims that ANO-CPA-secure KEM would be sufficient.

⁷ We refer the reader to Definition 3.

Practical Contribution: Implementation and Experimental Evaluation. We give an implementation of the protocol, clarifying all aspects of real-world deployment of the protocol, and we confirm experimentally the efficiency properties of the protocol. Our implementation builds on the reference implementation of Kyber—the full construction offering CCA security [9, 29] and anonymity [16, 21, 30]. We clarify how to instantiate the m2F components showing, in particular, that hashing into the public-key space of Kyber can be done by reusing the code that the Kyber created for expanding the seed ρ in the public key to a matrix over the algebraic ring that underlies the KEM construction. Technically, this entails proving that the rejection sampling procedure specified by Kyber is indifferentiable from a random oracle; a result that may be of independent interest. Compared to EKE-KEM [28], CHIC saves 32 bytes in bandwidth costs, while also bringing mild computational savings by (1) eliminating the need for Alice to generate 32 bytes of random coins, and (2) simplifying the inputs/outputs of the m2F construction, with the right wire carrying only part of the public key. The implementation is available as supplementary material.

2 Preliminaries

In this section, we present the definition of Key Encapsulation Mechanism (KEM) and introduce its security properties of interest for this work.

Definition 1. *A Key Encapsulation Mechanism (KEM) scheme is a tuple of polynomial-time algorithms $\text{KEM} = (\text{Keygen}, \text{Encap}, \text{Decap})$ that behaves as follows:*

- $\text{Keygen}(\lambda) \rightarrow (pk, sk)$: a key-generation algorithm that on input a security parameter λ , outputs a public/private key pair (pk, sk) .
- $\text{Encap}(pk) \rightarrow (c, K)$: an encapsulation algorithm that on input a public key pk , generates a ciphertext c and a secret key K .
- $\text{Decap}(sk, c) \rightarrow K$: a decapsulation algorithm that on input a private key sk and a ciphertext c , output a secret key K .

For correctness, we require that for any key pair $(pk, sk) \leftarrow \text{Keygen}(\lambda)$, and ciphertext and secret key $(c, K) \leftarrow \text{Encap}(pk)$, we have that $K = \text{Decap}(sk, c)$.

KEM Security Properties. The standard security notion for key encapsulation mechanisms and public-key encryption in general is indistinguishability under chosen-ciphertext attacks (IND-CCA). In addition to achieving IND-CCA security, many applications also demand the property of anonymity in a KEM. An anonymous KEM ensures that a ciphertext conceals the identity of the recipient by revealing no information about the public key employed in the encapsulation process. Unsurprisingly, the standard definition of this property is a logical adaptation of IND-CCA known as ‘anonymity under chosen-ciphertext attacks’ (ANO-CCA), and with the adversary’s objective being to determine which of two public keys was used to generate the given challenge ciphertext.

It has been widely established that Kyber is IND-CCA [9, 29] and ANO-CCA [16, 21, 30]. However, for our construction, we rely on weaker properties, namely ‘one-wayness under plaintext-checkable attacks’ (OW-PCA)⁸ and anonymity under plaintext-checkable attacks (ANO-PCA). In addition, we require a KEM with ‘splittable and pseudo-uniform public-keys’ (UNI-PK). All these notions are defined in Fig. 1.

Exp ANO-iPCA _{KEM} ^A (λ)	Exp OW-iPCA _{KEM} ^A (λ)
$(pk_0, sk_0) \leftarrow \text{Keygen}(\lambda)$	$(pk, sk) \leftarrow \text{Keygen}(\lambda)$
$(pk_1, sk_1) \leftarrow \text{Keygen}(\lambda)$	$(c^*, _) \leftarrow \text{Encap}(pk)$
$b \leftarrow_{\$} \{0, 1\}$	$K \leftarrow \mathcal{A}^{\text{PCO}_{i, \perp}(\text{sk}, \cdot)}(pk, c^*)$
$(c^*, _) \leftarrow \text{Encap}(pk_b)$	return $K == \text{Decap}(sk, c^*)$
$b' \leftarrow \mathcal{A}^{\text{PCO}_{i, c^*}(\text{sk}_0, \cdot)}(pk_0, pk_1, c^*)$	
return $b == b'$	
<hr/>	<hr/>
Oracle $\text{PCO}_{i, c^*}(\text{sk}, c, K)$	Exp UNI-PK _{KEM, Split} ^A (λ)
<hr/>	<hr/>
[\mathcal{A} can only make i queries]	$(pk_0, _) \leftarrow \text{Keygen}(\lambda)$
if $c == c^*$ return \perp	$(r_0, M_0) \leftarrow \text{Split}(pk_0)$
return $K == \text{Decap}(\text{sk}, c)$	$(r_1, M_1) \leftarrow N_\lambda \times G_\lambda$
	$b \leftarrow_{\$} \{0, 1\}$
	$b' \leftarrow \mathcal{A}(r_b, M_b)$
	return $b == b'$

Fig. 1. Security experiments defining properties of KEM: (1) One-Wayness under Plaintext-Checkable Attacks (OW-iPCA); (2) Anonymity under Plaintext-Checkable Attacks (ANO-iPCA); (3) Splittable and pseudo-Uniform Public-Keys (UNI-PK). \mathcal{A} is restricted to making at most i queries to the plaintext-checking oracle PCO. In particular, if $i = 0$, the plaintext-checking oracle PCO is not available. In ANO-iPCA security experiment, \mathcal{A} is not allowed to query the plaintext-checking oracle PCO on the challenge ciphertext c^* . This restriction is *not* imposed in OW-iPCA experiment.

To elaborate, we first adopt the notion of one-wayness under plaintext checkable attacks from [26]. We consider an adversary whose goal is to decrypt a KEM ciphertext without the private decapsulation key but with access to a plaintext-checking oracle. This oracle allows the adversary to confirm if the decapsulation of a ciphertext under the challenge decryption key corresponds to a particular plaintext (i.e., the secret key K in the context of KEM).⁹

⁸ Our construction can be proven secure assuming the underlying KEM has only ‘one-wayness under chosen-plaintext attacks’ (OW-CPA), though the proof is less tight.

⁹ In IND-CCA security game, the decapsulation oracle can be queried on anything except the challenge ciphertext. However, in OW-PCA, the plaintext-checking oracle is unrestricted. Despite this gap, one can show that IND-CCA implies OW-PCA with a tight reduction, losing only a constant factor 2. For the proof, we refer the reader to the full version of the paper [4].

Definition 2 (KEM one-wayness under plaintext-checkable attacks). A Key Encapsulation Mechanism (KEM) scheme is said to be OW-iPCA secure if for any PPT adversary \mathcal{A} engaged in the OW-iPCA security game, where \mathcal{A} is restricted to making at most i queries to the plaintext-checking oracle PCO, the advantage of \mathcal{A} defined as:

$$\text{Adv}_{\text{KEM}, \mathcal{A}}^{\text{OW-iPCA}}(\lambda) \stackrel{\text{def}}{=} \Pr[\text{OW-iPCA}_{\text{KEM}}^{\mathcal{A}}(\lambda) = 1] \quad (1)$$

is a negligible function of the security parameter λ . Experiment OW-iPCA is defined in Fig. 1.

Similarly, we also adopt a weaker variant of the ANO-CCA property, where the decapsulation oracle is replaced with the less-capable plaintext-checking oracle. We call this definition ANO-PCA. In ANO-CCA game, the decryption oracle disallows queries on the challenge ciphertext. We preserve this restriction in ANO-PCA so it is trivial to see that ANO-CCA implies ANO-PCA with no tightness loss in the reduction, as the plaintext-checking oracle could be easily simulated with a decapsulation oracle.

Definition 3 (KEM anonymity under plaintext-checkable attacks). A Key Encapsulation Mechanism (KEM) scheme is said to be ANO-iPCA secure if for any PPT adversary \mathcal{A} engaged in the ANO-iPCA security game, where \mathcal{A} is restricted to making at most i queries to the plaintext-checking oracle PCO and is prohibited from calling the oracle on the challenge ciphertext, the advantage of \mathcal{A} defined as:

$$\text{Adv}_{\text{KEM}, \mathcal{A}}^{\text{ANO-iPCA}}(\lambda) \stackrel{\text{def}}{=} 2 \cdot \Pr[\text{ANO-iPCA}_{\text{KEM}}^{\mathcal{A}}(\lambda) = 1] - 1 \quad (2)$$

is a negligible function of the security parameter λ . Experiment ANO-iPCA is defined in Fig. 1.

Finally, a less common security requirement but which proved to be essential for the constructions of PAKE protocol from KEM and IC [3, 5, 27, 28] is public key indistinguishability from uniform. In other words, the public keys output by the KEM key generation algorithm must be computationally indistinguishable from public keys uniformly sampled from the same key space. This notion is also known as fuzziness [5, 27]. In this work, we extended the requirements for KEM public keys. Namely, CHIC requires a KEM with splittable and uniform public keys that meet the following criteria: (1) the public key can be encoded as a bitstring and a group element; (2) a random oracle indifferentiable hash onto the group exists; and (3) honestly generated and decomposed public keys appear uniformly distributed.¹⁰

¹⁰ Later in our construction, the KEM public key will be split into two parts. The first part will be expanded to match the range of the second part—which is an element of a group—and used to mask the second part of the public key. The expansion function will be treated as a random oracle in our security proof. While it is gen-

Definition 4 (KEM with splittable and pseudo-uniform public keys).

A KEM scheme has splittable and pseudo-uniform public keys if (1) there exists an efficiently computable and invertible map $\text{Split} : \mathcal{PK}_\lambda \rightarrow N_\lambda \times G_\lambda$, such that each security parameter λ defines domains \mathcal{PK}_λ , G_λ , and $N_\lambda = \{0, 1\}^{p(\lambda)}$ for some polynomial p ; (2) there exists an RO-indifferentiable hash from N_λ onto G_λ ; (3) for any PPT adversary \mathcal{A} engaged in the UNI-PK security game, the advantage of \mathcal{A} defined as:

$$\text{Adv}_{\text{KEM, Split}, \mathcal{A}}^{\text{UNI-PK}}(\lambda) \stackrel{\text{def}}{=} 2 \cdot \Pr[\text{UNI-PK}_{\text{KEM, Split}}^{\mathcal{A}}(\lambda) = 1] - 1 \quad (3)$$

is a negligible function of the security parameter λ . Experiment UNI-PK is defined in Fig. 1.

PQ KEMs with Splittable and Uniform Public Keys. Crystals-Kyber [29] public key consists of a seed $\rho \in \{0, 1\}^{256}$ and a group element $\mathbf{t} \in R_q^k$, where R_q is the ring $\mathbb{Z}_q[X]/(X^n + 1)$, $q = 3329$ is a small prime, $n = 256$ and $k \in \{2, 3, 4\}$ depending on the choice of the security parameter λ . Consider the split algorithm for Kyber KEM to be the trivial breakdown of Kyber public keys into these two components. Seed ρ is derived from expanding a purely random bitstring $d \in \mathbb{B}^{32}$ using a hash function $G(d)$ that produces two 32-byte outputs, with ρ being one of them. In the security proofs of Kyber [16, 21, 29, 30], function G is modeled as a random oracle, which ensures that the distribution of ρ is uniform. In FIPS 203 [25] standard, function G is specified to be instantiated as SHA3-512. ρ is then further expanded into a large stream of candidate 12-bit values via an eXtendable Output Function (XOF), which Kyber instantiates with SHAKE-128. From this stream, the first candidate values within the range $[0, 3329)$ are selected to form the public matrix $\mathbf{A} \in R_q^{k \times k}$ in the NTT domain. This process is known as *rejection sampling* and, again, can be modelled as a random oracle mapping ρ to \mathbf{A} . Matrix \mathbf{A} is then used to compute the second component of the public key as a Module-LWE instance. Therefore, the Kyber public key can be shown to be pseudo-uniform under the decisional MLWE assumption [5] in the Random Oracle Model. To instantiate the RO that maps elements in $N_\lambda = \{0, 1\}^{256}$ to $G_\lambda = R_q^k$, we borrow the same rejection sampling procedure from Kyber. We expand this intuition and show that Kyber has splittable and pseudo-uniform public keys in the full version of this paper [4].

Other lattice-based KEMs employ the same technique of expanding a short seed into a public matrix, making them good candidates for splittable keys. For

erally accepted that random oracles with fixed ranges can be easily instantiated with cryptographically-secure hash functions, the instantiation of a random oracle for hashing into the group where (part of) the KEM public keys reside is less straightforward. Indifferentiability [22] allows to formally justify the instantiation of a non-trivial hashing procedure: it ensures that one can safely replace an ideal object (e.g., a RO that hashes into a group) with a construction that makes use of another ideal object (e.g., an ideal eXtendable Output Function). The requirement (2) emphasizes the need for such a hashing procedure to safely instantiate our protocol.

example, FrodoKEM [24], the lattice-based KEM recommended by the German Federal Office for Information Security (BSI)¹¹, also has splittable and pseudo-uniform public keys. Similarly to Kyber, FrodoKEM public keys can be trivially decomposed into a seed $seed_A \in \{0, 1\}^{128}$ (that expands to an $n \times n$ matrix \mathbf{A} , where all the coefficients are in \mathbb{Z}_q), and a group element $\mathbf{B} \in \mathbb{Z}_q^{n \times 8}$. The instantiation of the RO-indifferentiable hash-onto-group is even simpler for FrodoKEM because q is required to be a power of 2, so rejection sampling is not needed.

CPA Versus iPCA. Some essential points should be noted concerning these security definitions. Firstly, when access to the PCO oracle is restricted to zero queries, it effectively results in the removal of the oracle from the experiment. This, in turn, gives rise to the weaker definitional variants known as ‘chosen-plaintext attacks,’ specifically OW-CPA and ANO-CPA. Furthermore, we made two adjustments to weaken our ANO-iPCA definition: (a) we refrained from providing the adversary with the challenge secret key K^* , and (b) we restricted the PCO oracle to queries on the left private decapsulation key sk_0 . This contrasts with definitions in [16, 21, 30], which grant the adversary access to both keys via the oracle. These adaptations, which relax the requirements of the underlying KEM, are proven to be sufficient for establishing the security of the protocol CHIC presented in this paper.

It is also worth mentioning that for a very limited number of queries to the PCO oracle, OW-iPCA is equivalent to OW-CPA, as established by Lemma 1. However, it is essential to recognize that this equivalence cannot be readily extended to indistinguishability-based games. In such games, a flawed simulation resulting from an incorrect coin flip could nullify the advantage gained when the simulation was correct. Consequently, we cannot make a similar assertion regarding the relationship between ANO-iPCA and ANO-CPA.

Lemma 1. *If KEM is a OW-CPA secure key encapsulation mechanism, then it is also OW-1PCA secure.*

Proof. Let \mathcal{A} be any adversary against game OW-1PCA. We construct an adversary \mathcal{B} against OW-CPA that simulates game OW-1PCA for \mathcal{A} as follows: i. Challenge (pk, c^*) is forwarded to \mathcal{A} . ii. The single oracle query to PCO is answered by \mathcal{B} with a coin flip. iii. Finally, \mathcal{B} forwards \mathcal{A} ’s answer to OW-1PCA as its own answer to OW-CPA.

Notice that \mathcal{B} perfectly simulates OW-1PCA for \mathcal{A} half of the time, no matter what is \mathcal{A} ’s strategy for querying the plaintext-checking oracle. Therefore, at least half of the time (possibly more, in case \mathcal{A} wins regardless of the bad simulation of PCO), a win for \mathcal{A} translates into a win for \mathcal{B} .

$$\text{Adv}_{\text{KEM}, \mathcal{B}}^{\text{OW-CPA}}(\lambda) \geq \frac{1}{2} \cdot \text{Adv}_{\text{KEM}, \mathcal{A}}^{\text{OW-1PCA}}(\lambda) \quad (4)$$

In broader terms, OW-iPCA is essentially equivalent to OW-CPA, but only when the number of queries made to the plaintext-checking oracle is limited to

¹¹ BSI TR-02102-1, Version: 2024-1.

a few, as attempting to guess the PCO oracle's responses multiple times leads to an exponential loss in the number of tosses.

Definition 5 (Modified 2-Feistel construction: m2F). *The modified 2-round Feistel network, as introduced in [28], is constructed using three components: (1) block cipher denoted by the tuple of algorithms (IC.Enc, IC.Dec), with key space K and input/output space N ; (2) hash function H whose output space is represented by group G ; and (3) hash function H' whose output space is K . The m2F construction encompasses two efficiently computable functions, $\text{m2F}_{pw} : N \times G \rightarrow N \times G$ and its inverse m2F_{pw}^{-1} , both shown in Fig. 2.*

$\text{m2F}_{pw}(r, M)$	$\text{m2F}_{pw}^{-1}(s, T)$
$R \leftarrow H(pw, r)$	$t \leftarrow H'(pw, T)$
$T \leftarrow M \odot R$	$r \leftarrow \text{IC.Dec}(t, s)$
$t \leftarrow H'(pw, T)$	$R \leftarrow H(pw, r)$
$s \leftarrow \text{IC.Enc}(t, r)$	$M \leftarrow T \odot R^{-1}$
return (s, T)	return (r, M)

Fig. 2. The modified 2-Feistel [28], where \odot is a group G operation, and $(\cdot)^{-1}$ is an inverse in G .

Half-Ideal Cipher (HIC). In [28], the authors introduce a UC security notion they called (randomized) *Half-Ideal Cipher* (HIC), which is designed to relax the UC notion of an ideal cipher. This security notion is established through the introduction of an ideal functionality, denoted as \mathcal{F}_{HIC} , and is parameterized by the domain $N \times G$. Notably, \mathcal{F}_{HIC} features ‘honest’ interfaces accessible for queries by the environment \mathcal{Z} , with these queries being mediated through honest parties. However, the honest interfaces are restricted w.r.t. to half of the input: \mathcal{Z} has no control over the randomness parameter $r \in N$ in the encryption direction, and it cannot observe the value of r during decryption. By contrast, \mathcal{F}_{HIC} provides two adversarial interfaces that grant the adversary/simulator the capability to select r and even program half of the output $T \in G$ during encryption. In the decryption direction, the adversary can also observe the value of r .

It is shown in [28] that the m2F construction realizes \mathcal{F}_{HIC} functionality in the Random Oracle and Ideal Cipher (IC) model. The HIC abstraction serves as an effective replacement for an ideal cipher in the construction of EKE-like protocols, eliminating the need for the direct use of an IC over groups, whose instantiations are non-trivial (e.g., see [28]). However, it's worth noting that the randomized encryption of HIC introduces an overhead equal to the length of r . Due to the security proof of m2F requiring no collisions on the domain of the

IC, this overhead essentially amounts to 2λ bits, which is precisely what our construction CHIC optimizes.

HIC+ and Why It Fails. A natural question is whether the \mathcal{F}_{HIC} could be extended to $\mathcal{F}_{\text{HIC}+}$ which empowers honest parties and provides them with the ability to select and have visibility over r in respectively encryption and decryption. Unfortunately, the m2F construction would not be a provably secure realization of such extended functionality. To see why, let us exemplify with a concrete attack coordinated between an environment \mathcal{Z} and its adversary \mathcal{A} :

1. \mathcal{Z} selects r and M at random from the respective domains, picks arbitrary pw , queries $\mathcal{F}_{\text{HIC}+}$, via a honest party, on $\text{Enc}(pw, (r, M))$, and obtains some ciphertext (s, T) .
2. \mathcal{Z} queries $\text{H}'(pw, T)$ via its adversary \mathcal{A} and obtains t (a key for IC).
3. \mathcal{Z} queries $\text{IC.Enc}(t, r)$ via its adversary \mathcal{A} and should get back s .

Unfortunately, the simulator SIM cannot possibly know how to correctly answer the last query because it has no visibility over the first query \mathcal{Z} made to $\mathcal{F}_{\text{HIC}+}$, even though it controls IC, H and H'. This would not happen using \mathcal{F}_{HIC} interfaces because the environment \mathcal{Z} can only pass message M to the honest party Enc interface in step (1) above, and it would not know the randomness r (which in the real-world would be internally chosen by that honest party).

Although we could not leverage the modular abstraction that \mathcal{F}_{HIC} introduces (or an extension of it), we still take full advantage of the m2F construction, as a white-box drop-in, in our protocol CHIC and rely directly on the RO and IC in the security proof. Therefore, no security definition is formally introduced here for \mathcal{F}_{HIC} , and m2F is not explicitly parameterized by a security parameter λ , although its internal components are essential to the security analysis of CHIC.

3 Security Model

We begin this section with a brief review of the Universal Composability (UC) framework. Then we present the standard PAKE functionality as defined by Canetti et al. [13].

Let \mathcal{P} be a protocol of interest whose security properties are modelled within the UC framework. In this framework, the environment \mathcal{Z} embodies some higher-level protocol that uses \mathcal{P} as a sub-protocol, while also acting as an adversary attacking that higher-level protocol. Here, the adversary \mathcal{A} represents the adversary attacking protocol \mathcal{P} . Between the environment \mathcal{Z} and the adversary there is a continuously open communication channel. Such setup allows \mathcal{Z} to launch an attack on the higher-level protocol with the help of \mathcal{A} (who is attacking protocol \mathcal{P}). Note that \mathcal{Z} can only indirectly (through adversary \mathcal{A}) make calls to idealized primitives such as an Ideal Cipher and/or a Random Oracle.

In the UC framework that models the security of PAKE protocols, parties are initialized by the environment \mathcal{Z} with arbitrary passwords of the environment's choice. In the real world, protocols are executed according to protocol

specifications, in the presence of an adversary \mathcal{A} capable of dropping, injecting, and modifying protocol messages at will, thus modelling an insecure network. In the ideal world, parties do not execute the protocol. Instead, they interact via an ideal functionality $\mathcal{F}_{\text{PAKE}}$ described in Fig. 3, in the presence of a simulator SIM that acts as an adversary operating in the ideal world. The simulator SIM is also allowed to interact with $\mathcal{F}_{\text{PAKE}}$, but only using the $\mathcal{F}_{\text{PAKE}}$ adversarial interfaces as defined in Fig. 3.

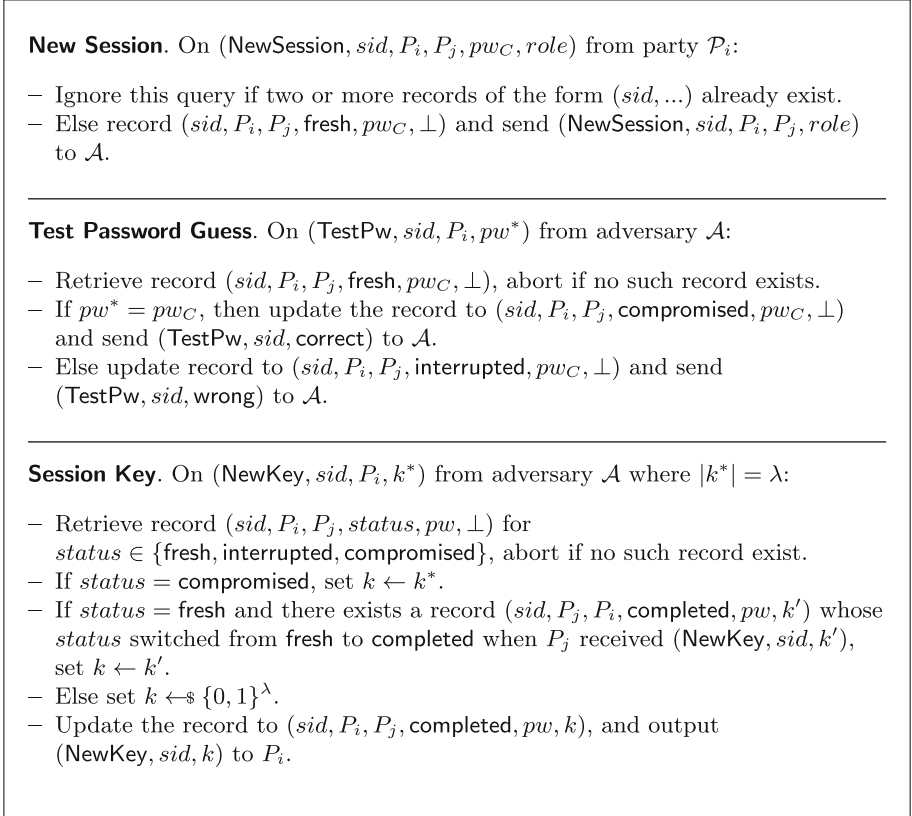


Fig. 3. The PAKE ideal functionality $\mathcal{F}_{\text{PAKE}}$ of Canetti et al. [13].

Finally, the goal of the environment \mathcal{Z} that interacts with the parties and the adversary (either real world \mathcal{A} or ideal world SIM) is to guess if it is in the real or in a simulation of the ideal world. Consequently, if for every efficient adversary \mathcal{A} no such efficient environment \mathcal{Z} exists that distinguishes the real world from the ideal world, we say that the protocol of interest \mathcal{P} securely emulates ideal functionality $\mathcal{F}_{\text{PAKE}}$. The UC PAKE definition results in a stronger notion than game-based PAKE notions and successfully captures the scenario where clients

register related passwords with different servers, as this is captured by the ability of \mathcal{Z} initializing parties with passwords of its choosing. Furthermore, the UC framework also ensures security under arbitrary protocol composition. Note that the environment \mathcal{Z} may reveal various information to the adversary \mathcal{A} , thus allowing UC PAKE definitions to capture password leaks (static adversaries) and internal state leaks (adaptive adversaries) that may occur anytime during the protocol execution.

4 UC PAKE from Modified 2-Feistel and KEM

In this section, we present CHIC, a UC-secure Password Authenticated Key Exchange protocol. CHIC assumes a KEM scheme that is one-way secure (at least OW-CPA, but OW-PCA results in a tighter proof), anonymous (ANO-1PCA), and has splittable and pseudo-uniform public keys (UNI-PK). The protocol, shown in Fig. 4, is built upon the modified 2-Feistel (m2F) construction of Dos Santos et al. [28]. We take a moment to discuss several design choices in our protocol, which follows an EKE-style construction combining a KEM and the m2F.

First, a pivotal decision, in contrast to the strategy in [28], was to ‘de-randomize’ the m2F. We split the KEM public key and use the parts as inputs to the m2F. This approach helps us avoid employing an ideal cipher over a group, which can be both costly and challenging to instantiate and eliminates any communication overhead associated with the HIC abstraction.

Second, the inputs used for tag and session key generation realized through the H_1 and H_2 function calls in CHIC, are identical. This allows us to optimize our implementation by making a single call to a hash function with an extended output size of 2λ . Subsequently, the output is cut in two halves, one forming the tag and the other the session key.

Third, the password is exclusively used in the m2F construction and is not provided as input to either H_1 or H_2 . This design choice means that the initiator does not need to store the input password in memory while waiting for the responder’s answer. This choice has potential benefits in the event of a complete compromise of the initiator (including leakage of its internal state), as an attacker would be required to perform an offline dictionary attack to retrieve the initiator’s password under such circumstances. (However, we don’t analyse the security of our protocol under adaptive attacks in the UC sense, and this sort of attack scenario is not captured by the $\mathcal{F}_{\text{PAKE}}$ functionality).

Fourth, it is worth noting that in our protocol the initiator, instead of aborting, outputs a random session key in the event that the received tag is invalid. We opt for this approach to ensure that our construction aligns with the security requirements specified in the standard UC PAKE functionality from [13] that foresees implicit authentication. However, in practice, when implementing the protocol, it is possible for the initiator to abort in that case, thus achieving explicit responder-to-initiator authentication. Furthermore, it is assumed that protocol participants erase any internal state as soon as it becomes unnecessary for the execution of the protocol. This means that the initiator instance

after computing and sending apk erases its entire internal state (including the password) except $fullsid$, apk , pk , and sk .

Note that if function Split can be randomized, specifically if $\text{Split}(pk)$ returns (r, pk) for and $\text{Split}^{-1}(r, pk)$ returns pk , then the Split+m2F block in protocol CHIC would instantiate the randomized Half-Ideal Cipher construction of [28]. In that sense, the Split+m2F procedure used in CHIC can be seen as a strict generalization of the HIC construction of [28].

5 Security Analysis

In this section, we prove that the protocol described in Fig. 4 UC-realizes the standard PAKE functionality $\mathcal{F}_{\text{PAKE}}$ shown in Fig. 3.

Theorem 1. *Let KEM be a OW-CPA, ANO-1PCA, and UNI-PK-secure key encapsulation mechanism. Let IC be a block cipher modeled as an ideal cipher, and H, H', H_1 and H_2 be hash functions modeled as random oracles. Then, the PAKE protocol CHIC described in Fig. 4 UC-realizes $\mathcal{F}_{\text{PAKE}}$ in the static corruption model. Furthermore, a OW-PCA-secure KEM leads to a tighter proof.*

Proof Overview. To prove Theorem 1 we show that the environment cannot distinguish between the “real world” experiment in which the environment \mathcal{Z} and adversary \mathcal{A} have parties P_i and P_j execute the protocol from Fig. 4, from an “ideal world” experiment in which a simulator SIM interacts with $\mathcal{F}_{\text{PAKE}}$ and presents to environment \mathcal{Z} a view that is consistent with what \mathcal{A} produces in the real world. We assume without loss of generality that \mathcal{A} is the dummy adversary, functioning as a communication intermediary between parties and the environment.

The Simulator. We describe the UC simulator SIM for CHIC that will act as the ideal-world adversary, having access to the ideal functionality $\mathcal{F}_{\text{PAKE}}$. SIM must simulate to \mathcal{Z} protocol messages between honest participants without knowing the passwords chosen by \mathcal{Z} , while consistently answering random oracle and ideal cipher queries. In a limited number of cases, the simulator is unable to conclude the simulation and aborts. We argue in the proof that those bad events only happen with negligible probability and account for these events in the overall probability of \mathcal{Z} distinguishing between the “real world” from the “ideal world”.

- *First message:* After receiving $(\text{NewSession}, sid, P_i, P_j, \text{Alice})$ from $\mathcal{F}_{\text{PAKE}}$, SIM picks a random apk and sends message apk from P_i to P_j .
- *Second message:* After receiving $(\text{NewSession}, sid, P_j, P_i, \text{Bob})$ from $\mathcal{F}_{\text{PAKE}}$, SIM waits for a message apk sent to P_j from \mathcal{A} . Then SIM sets $fullsid \leftarrow (sid, P_i, P_j)$. In case the received apk is an output of the m2F that commits the adversary to a password pw , SIM extracts the password pw and tests it by sending $(\text{TestPwd}, sid, P_j, pw)$ to $\mathcal{F}_{\text{PAKE}}$. If $\mathcal{F}_{\text{PAKE}}$ replies with “correct guess”, SIM computes the *key* according to the protocol specification and

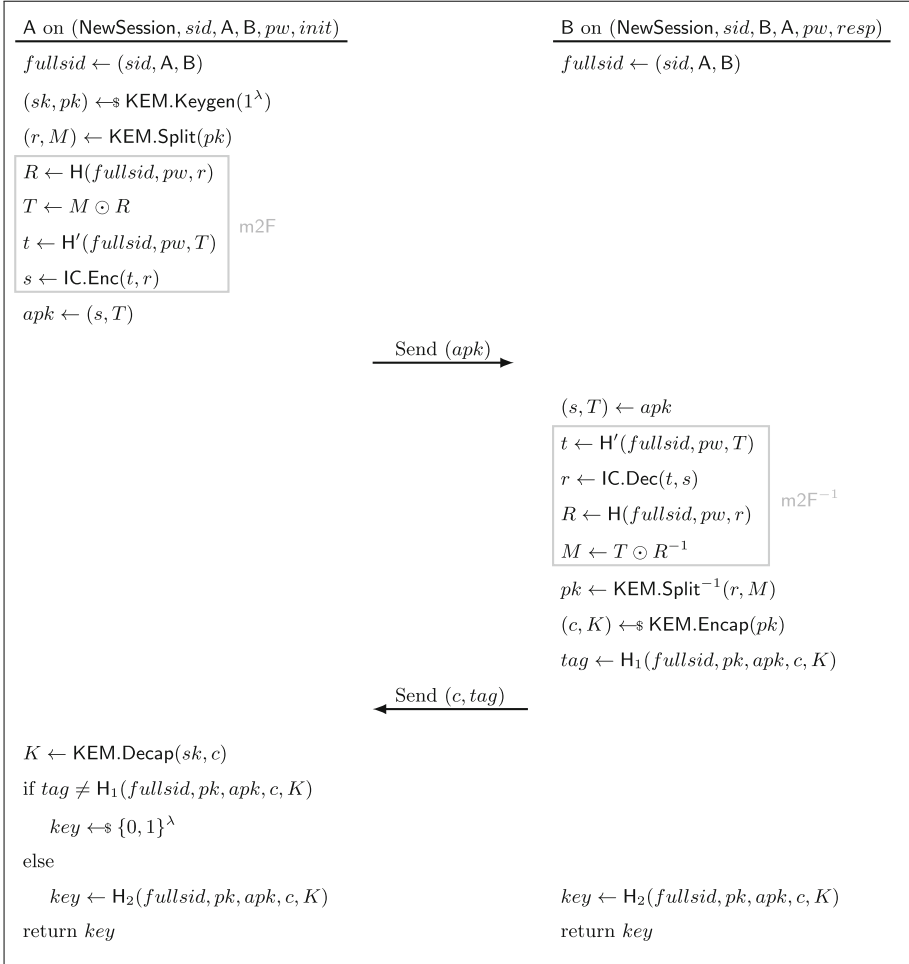


Fig. 4. The CHIC protocol. KEM scheme has splittable public keys (Definition 4) with an efficiently computable and invertible map $\text{Split} : \mathcal{PK}_\lambda \rightarrow N_\lambda \times G_\lambda$. The protocol makes use of a block cipher denoted as IC and hash functions H and H' in an m2F configuration (Definition 5), with domains that align with Split and that are characterized by security parameter λ , i.e. $\{\text{IC.Enc}, \text{IC.Dec}\} : K_\lambda \times N_\lambda \rightarrow N_\lambda$, $\text{H} : \{0, 1\}^* \rightarrow G_\lambda$, $\text{H}' : \{0, 1\}^* \rightarrow K_\lambda$. Group operations within G are represented by \odot , and the inverse operation by $(\cdot)^{-1}$.

sends $(\text{NewKey}, sid, P_j, key)$ to $\mathcal{F}_{\text{PAKE}}$; in all other cases (including “wrong guess”, honest execution, etc.), SIM runs a KEM.Keygen algorithm, obtains a fresh key pair (pk, sk) , computes the ciphertext c and the tag using the fresh pk , and sends $(\text{NewKey}, sid, P_j, \perp)$ to $\mathcal{F}_{\text{PAKE}}$. To conclude the second message flow, SIM sends the message (c, tag) from P_j to P_i via \mathcal{A} .

- *Final output:* After receiving message (c, tag) sent to P_i from \mathcal{A} , in case of honest execution, SIM simply sends $(\text{NewKey}, sid, P_i, \perp)$ to $\mathcal{F}_{\text{PAKE}}$. If message (c, tag) was tampered with by the adversary, SIM checks for a corresponding random oracle query to H_1 that returned tag . If such query has not been asked, SIM sends $(\text{TestPwd}, sid, P_i, \perp)$ and $(\text{NewKey}, sid, P_i, \perp)$ to $\mathcal{F}_{\text{PAKE}}$, forcing a random session key. If tag comes from H_1 , pk and apk are extracted. If appropriate queries were made to the m2F, the password is also extractable. SIM extracts \mathcal{A} 's password guess pw and sends $(\text{TestPwd}, sid, P_i, pw)$ to $\mathcal{F}_{\text{PAKE}}$. In case of a “correct guess”, SIM computes the key by following the protocol and sends $(\text{NewKey}, sid, P_i, key)$ to $\mathcal{F}_{\text{PAKE}}$. If tag is not valid (even if the password guess was correct) or $\mathcal{F}_{\text{PAKE}}$ returned “wrong guess”, SIM sends $(\text{NewKey}, sid, P_i, \perp)$ to $\mathcal{F}_{\text{PAKE}}$. If the adversary did not commit to a password in its interaction with m2F, SIM sends $(\text{TestPwd}, sid, P_i, \perp)$ and $(\text{NewKey}, sid, P_i, \perp)$ to $\mathcal{F}_{\text{PAKE}}$.

Proof. We prove Theorem 1 via a series of game hops. The first game corresponds to a simulator that is not constrained in any way and executes the real world for the environment perfectly. Concretely, this simulator controls all inputs/outputs to the parties, as well as their communications with the environment. In each hop, we modify this simulator gradually, so that in the final game one can clearly see that it can be divided into two parts, where the first part corresponds to the ideal functionality $\mathcal{F}_{\text{PAKE}}$ and the second part to the simulator described earlier, which has only black-box access to $\mathcal{F}_{\text{PAKE}}$ and does not know the honest parties secret passwords. Conceptually, we think of $\mathcal{F}_{\text{PAKE}}$ as always existing alongside our simulator and receiving the inputs from \mathcal{Z} : in the first game it is not used at all by the simulator, and gradually it will start using $\mathcal{F}_{\text{PAKE}}$ to define the outputs of parties. Because the first game is identical to the real world and the last game is identical to the ideal world, we just need to show that the view of the environment is not affected by each of our modifications. Hence, in each hop, we analyze the probability of \mathcal{Z} outputting 1 in the game G_i compared to that of \mathcal{Z} outputting 1 in the game G_{i-1} and show that these change by a negligible amount.

Our analysis depends on the number of interactions between the environment and the execution model. To account for this, we consider and tally all queries made to the ideal cipher and random oracles, irrespective of whether they originate from honest parties or the adversary. We denote q_{IC} as the upper bound on queries to the ideal cipher, regardless of whether it is used for encryption (IC.Enc) or decryption (IC.Dec). Similarly, q_H , $q_{H'}$, and q_{H_1} represent upper bounds on the number of queries made to the H , H' , and H_1 oracles, respectively. Furthermore, we take into account the number of PAKE sessions and interactions occurring within each session. In this context, $q_{\text{newSession}}$ serves as an upper bound on the number of sessions initiated by \mathcal{Z} , while q_{send} represents an upper bound on the number of messages delivered by \mathcal{A} when interacting with the involved parties.

Game G_0 (Real World): Simulation perfectly mimics the world with oracles H , H' , H_1 , H_2 , IC.Enc and IC.Dec.

$$\Pr[G_0] = \mathbf{Real}_{\mathcal{Z}, \mathcal{A}, \text{CHIC}} \quad (5)$$

Game G_1 (Abort on Random Oracle Collisions): On output collisions of H_1 , H or H' , the simulation aborts. This is a statistical hop with a birthday bound.

$$|\Pr[G_0] - \Pr[G_1]| \leq \frac{q_{H_1}^2}{|\text{Space}_{H_1}|} + \frac{q_H^2}{|\text{Space}_H|} + \frac{q_{H'}^2}{|\text{Space}_{H'}|} \quad (6)$$

Game G_2 (Full Domain Sampling of IC and Abort on Collisions): On new IC.Enc and IC.Dec queries, simulator samples s and r regardless of previous answers and instead aborts on output collisions (even collisions across different keys). s and r are high-entropy, therefore this is a statistical hop with a negligible difference. Note that queries must be answered consistently and thus decrypting a ciphertext returned by IC.Enc or encrypting a plaintext returned by IC.Dec, under the same key, is not considered a new query.

$$|\Pr[G_1] - \Pr[G_2]| \leq \frac{q_{\text{IC}}^2}{|\text{Space}_{\text{IC}}|} \quad (7)$$

Game G_3 (Abort If a New Sample for H' Collides with a Previous Record of the IC): Upon sampling a new t (key for ideal cipher) for the simulation of H' oracle, if t is not fresh (and therefore already included in List_{IC}), the simulation aborts. This is a statistical hop.

$$|\Pr[G_2] - \Pr[G_3]| \leq \frac{q_{H'} \cdot q_{\text{IC}}}{|\text{Space}_{H'}|} \quad (8)$$

Game G_4 (Abort If a New Sample for IC.Dec Collides with a Previous Record of H): Upon sampling a new r for the simulation of IC.Dec, if r is not fresh (and therefore already included in List_H), the simulation aborts. This is a statistical hop.

$$|\Pr[G_3] - \Pr[G_4]| \leq \frac{q_{\text{IC}} \cdot q_H}{|\text{Space}_{\text{IC}}|} \quad (9)$$

Game G_5 (On Calls to IC.Dec Where the Password is Extractable from the Ideal Cipher Key, Force a Record to H): On a new query IC.Dec(t, s)—i.e., a query where a fresh ideal cipher preimage r is sampled—check if t came out of H' oracle and, if so, introduce the following change to the oracle. First, extract the password pw associated with t (there is at most 1 since we have already discarded the possibility of collisions in H'), then call $H(pw, r)$, forcing r to be added into the records of oracle H . Note that due to the abort triggers introduced in the previous games, this modification is equivalent to sampling a random pair (r, R) and trying to program H directly by adding the tuple (pw, r, R) to List_H . This action will abort if either $(*, r, *) \in \text{List}_H$ (see Game G_4) or if $(*, *, R) \in \text{List}_H$ (see Game G_1). Nothing really changes unless IC.Dec

triggers an abort that did not occur in the previous game. This is a statistical hop.

$$|\Pr[G_4] - \Pr[G_5]| \leq \frac{q_{IC} \cdot q_H}{|\text{Space}_{IC}|} + \frac{q_{IC} \cdot q_H}{|\text{Space}_H|} \quad (10)$$

Game G_6 (On Calls to $IC.Dec$ Where the Password is Extractable from the Ideal Cipher Key, Use $KEM.Keygen$ and Store Secrets): On a new query $IC.Dec(t, s)$, if t came out of H' oracle, instead of directly sampling a random pair (r, R) , the simulator relies on $KEM.Keygen$ and $KEM.Split$, and stores the secrets in $List_{secrets}$ for future use. If the adversary attempts to decrypt Alice's apk using her password, the record is also added to $List_{secrets}$. More precisely, if the adversary queries $IC.Dec(t, s)$, where $t = H'(fullsid, pw, T)$ and $apk = (s, T)$ is the message Alice sent in the session matching $fullsid$, and pw is Alice's password for that session, then add (pw, sk, pk, apk) to $List_{secrets}$, where (sk, pk) is Alice's key pair. This hop is down to the uniformity of KEM public keys.

$$|\Pr[G_5] - \Pr[G_6]| \leq q_{IC} \cdot \text{Adv}_{KEM.Split}^{pk\text{-uniformity}} \quad (11)$$

Game G_7 (Set Random Key via \mathcal{F}_{PAKE} if tag was Not Output by H_1): Modify Alice's response when tag was not output by H_1 wrt $fullsid$, apk and c : use \mathcal{F}_{PAKE} to generate the session key totally at random by compromising the session with an invalid password and then completing the session with $NewKey$. The protocol specification determines Alice's session key to be random if tag is incorrect.¹² A tag not coming out of H_1 will only be valid with negligible probability. Therefore, this is a statistical hop.

$$|\Pr[G_6] - \Pr[G_7]| \leq \frac{q_{send}}{|\text{Space}_{H_1}|} \quad (12)$$

Game G_8 (for Passive Attacks, Use a Private Oracle H_1^* Without Inputs pk and K to Compute tag , and set Session Key Directly via \mathcal{F}_{PAKE} Instead of Using the Key Coming from H_2): For passive attacks, i.e. messages are correctly computed and forwarded to the intended party (apk from Alice to Bob, and possibly (c, tag) from Bob to Alice), compute the tag with private oracle H_1^* and use the functionality to generate the session key, without testing the password.

The intuition of this hop is that the KEM ciphertext must conceal K , therefore the adversary will not call $H_1(*, *, *, *, K)$ nor $H_2(*, *, *, *, K)$. If it does, the simulator breaks the one-wayness of the KEM . The technical difficulty in the reduction is that the simulator does not know ahead of time if the session will be actively attacked. Therefore, it must embed the challenge pk^* in each session, one at a time (hybrid argument), and complete the simulation without detectable changes to the protocol. If the adversary relays correctly apk from Alice to Bob but then decides to actively interfere with the communication and forward its own (c, tag) back to Alice, the simulator faces the dilemma of whether to force Alice to use a random session key (if tag is invalid) or the session key resulting

¹² Our protocol is implicitly rejecting to follow the standard \mathcal{F}_{PAKE} functionality.

from $H_2(\text{fullsid}, pk, apk, c, K)$ (if tag is valid). This boils down to whether c encrypts K included in tag or not. However, because we embedded the challenge pk^* to compute the first flow of messages, we no longer can decrypt c . For this reason, we reduce this hop down to OW-PCA and take advantage of the PCO oracle to check if the key K included in the tag is effectively the key encrypted under c .

The reduction goes as follows (hybrid argument, one public key at a time): i. Embed challenge pk^* into Alice’s initialization procedure. ii. If the adversary is passive and delivers apk to Bob, reduction uses challenge c^* and private oracles H_1^* and H_2^* to proceed. These private oracles receive the same inputs as their public counterparts H_1 and H_2 , except for the arguments pk and K . (Note that K^* encrypted under c^* is unknown to the reduction.) Since the ciphertext c^* is an input to both H_1 and H_2 , this fixes a single key anyway and the games are identical unless K^* is queried to either oracle. If such a query is never placed, the usage of these private oracles is independent of \mathcal{Z} ’s view. iii. On Alice’s side, if the adversary is still passive, decryption is not needed: tag is valid and session key is derived from private oracle H_2^* . Due to the uniqueness of inputs, private oracle H_2^* will produce the same key on both sides, as would the public oracle H_2 in G_7 and NewKey query to $\mathcal{F}_{\text{PAKE}}$ in G_8 .

If the adversary is active (and the reduction embedded the challenge pk^* in this session) the reduction algorithm will use the PCO oracle to verify the tag: it verifies that the unique H_1 entry corresponding to the tag includes key encrypted under c^* . In this reduction, there is at most one PCO call per embedded challenge public key pk^* since KEM decryption occurs only in one place in our protocol. If this check fails, the reduction returns a fresh random key to the attacker, which is consistent with both games: trivially so in G_7 , and in G_8 because this forces the functionality to produce a fresh random session key by issuing a TestPw with \perp .

When the adversary concludes its run, the reduction algorithm confirms the inclusion of the correct K^* in queries to H_1 and H_2 via calls to the PCO oracle before submitting its answer against the one-wayness property of KEM ciphertexts. If no such query with the correct K^* exists, G_7 and G_8 are identical. As an alternative approach, randomly selecting an entry from the H_1 and H_2 tables can lead to a less tight reduction. However, this approach does not require confirmation of the inclusion of K^* in H_1 and H_2 oracle queries, and therefore at most one query to the PCO oracle is required for the correct simulation of active attacks to Alice, after embedding the challenge public key in the first flow from Alice to Bob. Importantly, Lemma 1 establishes the equivalence between OW-1PCA and OW-CPA.

$$|\Pr[G_7] - \Pr[G_8]| \leq q_{\text{send}} \cdot \text{Adv}_{\text{KEM}}^{\text{ow-pca}} \quad (13)$$

$$|\Pr[G_7] - \Pr[G_8]| \leq (q_{H_1} + q_{H_2}) \cdot q_{\text{send}} \cdot \text{Adv}_{\text{KEM}}^{\text{ow-cpa}} \quad (14)$$

Game G_9 (Simulate Bob’s Response with a Fresh Public Key for Passive Attacks): For honestly transmitted apk , the simulator creates ciphertext c with a fresh public key, then computes tag with the private oracle H_1^* (as in

the previous game), and finally sends (c, tag) on Bob's behalf. We bridge this hop using ANO-1PCA, with a hybrid argument, replacing one public key at a time. Note that Alice does not decrypt honestly transmitted ciphertexts since G_8 . However, if there's an active attack on the second round of the session where the reduction programmed the challenge pk_0 from ANO-1PCA game, the decryption key is not available. As before, the reduction takes advantage of a single call to the PCO oracle and the (single) relevant record of H_1 to determine whether the tag is valid.

More in detail, the reduction goes as follows (hybrid argument, one public key at the time): i. Embed challenge pk_0 into Alice's initialization procedure. ii. If the adversary is passive and delivers apk to Bob, reduction uses challenge c^* . iii. On Alice's side, if the attack is passive, no need to decrypt c^* . If there is an active attack, extract K from tag by inspecting H_1 records, and check K against c submitted by the adversary and sk_0 to determine the validity of tag without actually decrypting the ciphertext. Note that the PCO oracle of the standard ANO-1PCA game allows checks against both sk_0 and sk_1 , and the reduction embedded pk_0 on Alice's side. Therefore, checks must be carried out against sk_0 .

There are a few noteworthy observations regarding the definitional requirements for this reduction. For starters, we only need a weaker version of ANO-1PCA where the PCO oracle only allows plaintext checks against one of the secret keys. Another observation is that we don't require the challenger of the ANO-1PCA game to provide K^* as part of the challenge. This is a direct result of the modification introduced in G_8 that lifts the need to use the encapsulated key for passive attacks (via the usage of private oracle H_1^* to compute the tag, and NewKey to $\mathcal{F}_{\text{PAKE}}$ to set the session key).¹³ This reduction algorithm perfectly interpolates between games G_8 and G_9 . If challenge c^* is a result of KEM.Encap with pk_0 , this corresponds to G_8 . On the other hand, if c^* is a result of KEM.Encap with pk_1 , the simulation adheres to the specifications of G_9 .

$$|\Pr[G_8] - \Pr[G_9]| \leq q_{\text{send}} \cdot \text{Adv}_{\text{KEM}}^{\text{ano-1pca}} \quad (15)$$

Game G_{10} (Active Attacks on Alice: The Tag is Invalid if the Password Cannot be Extracted from an Adversarially Crafted Message from Bob to Alice): On adversarially crafted (c, tag) sent to Alice, the tag forces a commitment to a single pk and, consequently, to a unique password due to the joint operation of IC.Dec and H' . The only case in which password extraction fails is if the adversary did not reconstruct the Pk to which it committed using calls to IC.Dec and H' . However, in this case, the correct Pk that Alice will be using is information-theoretically hidden from the adversary's view. More in detail, in G_{10} we check if the pk was not obtained from apk via the appropriate calls to H' and then IC.Dec. (Note that since G_6 the appropriate decryption calls create a record in $\text{List}_{\text{secrets}}$.) If this is not the case, then tag is declared as invalid. In such cases, we force a random session key via $\mathcal{F}_{\text{PAKE}}$.

¹³ We note that this is the point in the proof where we could not find a way to avoid a decryption-like oracle and that forces us to use an actively secure KEM.

The two games G_9 and G_{10} are identical unless the adversary guessed Alice’s public key (and created the *tag* sent to Alice with it) without having obtained it from *apk* via the appropriate calls to H' and then $IC.Dec$. The probability of guessing Alice’s public key by chance is tied to the min-entropy k of the public keys generated by $KEM.Keygen$. This is not a new assumption on the properties of KEM as it follows from $UNI-PK$ that each part of the split public key has at least min-entropy λ . We denote ϵ as a negligible function.

$$|\Pr[G_9] - \Pr[G_{10}]| \leq \epsilon(\lambda) \quad (16)$$

Game G_{11} (Active Attacks on Alice: If the Password is Extractable, Test it and Proceed Accordingly): On Alice’s side, if the password can be extracted, test the password via $TestPwd$. If the guess is correct, run the protocol honestly and program the session key. If the guess is wrong, tell functionality to complete the session with a random key. Note that different passwords are guaranteed to lead to different public keys for a fixed *apk*, as oracles discard collisions. In turn, because *pk* is also included as an argument of H_1 , the tag-verification procedure is bound to fail. Therefore, Game G_{10} and Game G_{11} are identical from \mathcal{Z} ’s perspective. This is a bridge hop.

$$\Pr[G_{10}] = \Pr[G_{11}] \quad (17)$$

Game G_{12} (Simulate Alice’s Initial Message Without Using the Password): Notice that the simulator deals with Alice’s response without using *sk*, except for the case where Alice is actively attacked with the correct password. Therefore, the simulator can simulate a $NewSession$ for Alice by directly sampling *apk*, leaving the generation of the public key for later. As such, the password *pw* is not required at this stage.

Nevertheless, the simulator of G_{12} creates an IC record for *apk* $:= (s, T)$, with placeholders that can later be replaced by Alice’s *pk*. More precisely, it adds $(\perp, \perp, s, mode = E)$ to $List_{IC}$. If s is unrefresh, the simulation aborts. The record only gets updated when Alice’s password *pw* is confirmed to be correct as a result of a $TestPwd$ query to \mathcal{F}_{PAKE} , and $m2F_{pw}^{-1}(apk)$ is computed by querying its oracles. Recall that queries to $IC.Dec$ with t that permits password extraction leads to *sk* being embedded in $List_{secrets}$. So, the decryption key is always available in the only case still needed (active attack with the correct password).

Following the rules of the previous game G_{11} , Alice generates a key-pair with $KEM.Keygen$ and then computes *apk* by feeding *pk* to the oracles of the $m2F$, which leads to early abortion if the newly sampled R is in $List_H$ (rule added in G_1), if the newly sampled t is in $List_{IC}$ (rule added in G_3), and if the newly sampled s is in $List_{IC}$ (rule added in G_2). In G_{12} , we abort only if the newly sampled s is in $List_{IC}$. This means that the other abortion events have to be accounted for in the analysis of this game hop. Furthermore, if the adversary places a new query to $IC.Enc$ and happens to land on s —it’s important to emphasize *new query*, meaning this only applies to queries $IC.Enc(t, r)$ where r is not the result of a previous query $IC.Dec(t, s)$,—the game aborts since there’s already a record (albeit incomplete). Notice that in G_{11} there is one particular value of

r for which the oracle will respond without aborting—this is the r obtained by splitting Alice’s pk . The probability of guessing r by chance is tied to the min-entropy k of the first element resulting from the split of public keys generated by KEM.Keygen . Again, this is not a new assumption about KEM, as it follows from UNI-PK.

$$|\Pr[G_{11}] - \Pr[G_{12}]| \leq \frac{q_{\text{newSession}} \cdot q_H}{|\text{Space}_H|} + \frac{q_{\text{newSession}} \cdot q_{H'}}{|\text{Space}_{H'}|} + \epsilon(\lambda) \quad (18)$$

Game G_{13} (Active Attacks on Bob: If There’s no Record Consistent with apk Having Been Computed in the Forward Direction, Use Private Oracle H_1^* to Compute tag an Set Random Session Key via $\mathcal{F}_{\text{PAKE}}$):

The attacker sends its own apk to Bob and there is no record consistent with apk having been computed in the forward direction. The term *forward direction* refers to the encryption direction within the ideal cipher, which is why the simulator tracks how the records of the IC were generated. Formally, $apk = (s, T)$ was computed in the forward direction if there exist a record $(t, *, s, mode = E) \in \text{List}_{\text{IC}}$ such that there is a record $(fullsid, *, T, t) \in \text{List}_{H'}$ wrt the unique $fullsid$ associated with the Alice instance receiving apk . In such cases, the simulator uses the private oracle H_1^* to compute the tag and sets a random session key via $\mathcal{F}_{\text{PAKE}}$. Recall that the private oracle H_1^* does not take as input pk and K . We reduce this hop down to OW-PCA. We use a hybrid argument, changing the behavior of one Bob session at a time. The intuition is that if apk was not computed in the forward direction with an appropriate call to IC.Enc , the attacker has no control over the KEM public key (and corresponding secret key) associated with apk sent to Bob. Therefore, the attacker cannot decrypt Bob’s ciphertext, and is unlikely to query H_1 with K encrypted in Bob’s response. If it does, we break the OW-PCA game of KEM.

The reduction algorithm knows Bob’s password. The inverse of the attacker’s apk sent to Bob, under Bob’s password, must be the challenge pk^* of the OW-PCA game. The difficulty in arguing this hop arises from the adversary’s potential actions with apk : they might attempt to decrypt it using Bob’s password before or after sending it, or they may not decrypt apk with Bob’s password at all (willingly or because the Bob’s password was never correctly guessed by the adversary).

Remember, in this particular game hop, we are exclusively handling adversary-generated apk values, which are not computed following the forward direction of the m2F. Therefore, we apply a hybrid argument over all q_{send} queries from Alice to Bob, and all IC.Dec queries, carefully associating the challenge pk^* with one of these queries. The reduction algorithm loses the ability to decrypt ciphertexts encrypted under pk^* , but in the protocol only Alice needs to decrypt ciphertexts and she will do so under her secret key (regardless of whether apk sent out is crafted by the adversary and possibly associated with pk^*).

The reduction algorithm also embeds c^* in the computation of tag with private oracle H_1^* and in Bob’s response. It also monitors queries to public oracles H_1 and H_2 , extracting K and testing with the PCO oracle against challenge c^* . If the PCO oracle returns *true*, the reduction would submit K and would win

the OW-PCA game. Otherwise, the usage of private oracle H_1^* and setting Bob's session key to be random via $\mathcal{F}_{\text{PAKE}}$ is identical from \mathcal{Z} 's view.

Alternatively, as also described in the proof strategy of the hop to G_8 , if we are willing to bear the cost of a loss in tightness, we could use a guessing argument instead by simply outputting a K queried to one of the public oracles H_1 and H_2 , and avoid relying on any PCO oracle for this reduction (as mentioned earlier, Alice is always able to decrypt).

$$|\Pr[G_{12}] - \Pr[G_{13}]| \leq (q_{\text{send}} + q_{\text{IC}}) \cdot \text{Adv}_{\text{KEM}}^{\text{ow-pca}} \quad (19)$$

$$|\Pr[G_{12}] - \Pr[G_{13}]| \leq (q_{H_1} + q_{H_2}) \cdot (q_{\text{send}} + q_{\text{IC}}) \cdot \text{Adv}_{\text{KEM}}^{\text{ow-cpa}} \quad (20)$$

Game G_{14} (Active Attacks on Bob: If There's no Record Consistent with apk Having Been Computed in the Forward Direction, Encrypt the Ciphertext Under a Freshly Generated Public Key): As in the case of the previous game hop, the attacker sends its own apk to Bob and there's no record consistent with apk having been computed in the forward direction. Now, the simulator encrypts the ciphertext that Bob sends out under a freshly generated public key. This is a reduction to ANO-CPA.

The reduction is similar to the previous game hop in that we embed pk_0 in one send query to Bob at the time, and then embed the challenge c^* in Bob's response. As in the analysis of the previous game hop, we have to account for the possibility that the attacker tried to decrypt apk under Bob's password before sending it. In that case, pk_0 needs to be embedded upon the IC.Dec call. In the worst case, the loss in tightness w.r.t. to ANO-CPA is limited by $q_{\text{send}} + q_{\text{IC}}$. If c^* was encrypted under pk_0 , we adhere to the specifications of G_{13} . If it was encrypted under pk_1 , we adhere to the rules of G_{14} . We have already established in the previous game that tag is computed via private oracle H_1 (that does not take pk as input). As in the reduction strategy of the previous game hop, the challenge pk^* of ANO-CPA is never associated with the apk sent by Alice. Thus, Alice's decryption key sk is always available when needed, and a PCO oracle is also not needed for this reduction.

$$|\Pr[G_{13}] - \Pr[G_{14}]| \leq (q_{\text{send}} + q_{\text{IC}}) \cdot \text{Adv}_{\text{KEM}}^{\text{ano-cpa}} \quad (21)$$

Game G_{15} (Active Attacks on Bob: If There is a Record Consistent with apk Having Been Computed in the Forward Direction, Extract the Password, Test it, and Use Private Oracle H_1^* and set a Random Session Key If "wrong guess"): The simulator now deals with the case where there is a record consistent with apk having been computed in the forward direction. The simulator extracts the password and tests it. If "correct guess", the simulator keeps following the protocol and sets the correctly-computed session key via $\mathcal{F}_{\text{PAKE}}$ (this doesn't change anything from \mathcal{Z} 's view). If "wrong guess", the simulator makes use of private oracle H_1^* to compute the tag and sets a random session key via $\mathcal{F}_{\text{PAKE}}$. The reduction is similar to that of G_{13} .

$$|\Pr[G_{14}] - \Pr[G_{15}]| \leq (q_{\text{send}} + q_{\text{IC}}) \cdot \text{Adv}_{\text{KEM}}^{\text{ow-pca}} \quad (22)$$

$$|\Pr[G_{14}] - \Pr[G_{15}]| \leq (q_{H_1} + q_{H_2}) \cdot (q_{\text{send}} + q_{IC}) \cdot \text{Adv}_{\text{KEM}}^{\text{ow-cpa}} \quad (23)$$

Game G_{16} (Active Attacks on Bob: If There is a Record Consistent with apk Having Been Computed in the Forward Direction, Extract the Password, Test It, and Encrypt the Ciphertext Under a Freshly Generated Public Key If “wrong guess”): This change and reduction is similar to that argued in G_{14} .

$$|\Pr[G_{15}] - \Pr[G_{16}]| \leq (q_{\text{send}} + q_{IC}) \cdot \text{Adv}_{\text{KEM}}^{\text{ano-cpa}} \quad (24)$$

Game G_{17} (Ideal World): At this point, we are in the ideal world, where the simulator is using the ideal functionality $\mathcal{F}_{\text{PAKE}}$ to generate all keys except for those where there is a correct password guess.

$$\Pr[G_{16}] = \Pr[G_{17}] = \text{Ideal}_{\mathcal{Z}, \text{SIM}, \mathcal{F}_{\text{PAKE}}} \quad (25)$$

Bringing all these elements together and assuming KEM is a OW-PCA-secure key encapsulation mechanism, we obtain the result shown in Eq. 26. For the sake of completeness, a description in pseudo-code of the simulator SIM of the ideal world is provided in the full version of the paper [4]. Each step of the process, starting from the code execution of uncorrupted parties in the real world and leading to the simulation of the ideal world, is meticulously detailed. Every modification is framed and cross-referenced with the specific game hop where it was initially introduced to ensure a traceable progression of the proof.

$$\begin{aligned} & |\text{Real}_{\mathcal{Z}, \mathcal{A}, \text{CHIC}} - \text{Ideal}_{\mathcal{Z}, \text{SIM}, \mathcal{F}_{\text{PAKE}}}| \leq \\ & \quad q_{IC} \cdot \text{Adv}_{\text{KEM}, \text{Split}}^{\text{pk-uniformity}} \\ & + (3 \cdot q_{\text{send}} + 2 \cdot q_{IC}) \cdot \text{Adv}_{\text{KEM}}^{\text{ow-pca}} \\ & + (3 \cdot q_{\text{send}} + 2 \cdot q_{IC}) \cdot \text{Adv}_{\text{KEM}}^{\text{ano-1pca}} \\ & + \frac{q_{IC}^2 + 2 \cdot q_{IC} \cdot q_H}{|\text{Space}_{IC}|} + \frac{q_{H_1}^2 + q_{\text{send}}}{|\text{Space}_{H_1}|} \\ & + \frac{q_H^2 + q_{IC} \cdot q_H + q_{\text{newSession}} \cdot q_H}{|\text{Space}_H|} \\ & + \frac{q_{H'}^2 + q_{IC} \cdot q_{H'} + q_{\text{newSession}} \cdot q_{H'}}{|\text{Space}_{H'}|} + \epsilon(\lambda) \end{aligned} \quad (26)$$

□

On Tightness. The bounds we give here are aligned with those obtained in prior works on EKE-like constructions from KEMs. The main difference with respect to Diffie-Hellman based constructions is that we cannot use self reducibility properties to remove the multiplicative factors associated with dealing with multiple-instance KEM security properties. Intuitively, the q_{IC} multiplicative factor is the most problematic, but it seems intrinsic to the use of the ideal cipher: it corresponds to the reduction’s uncertainty as to which of the adversary’s reverse

ideal cipher queries will the adversary choose to fix the KEM public key on which it will be challenged. A KEM with a tight proof of multi-instance security would solve this problem.

Implications for the Proofs in [5,28]. In game G_9 we explain why at that point in the proof we could not avoid making a decryption-like query, and that forces us to use an actively secure KEM. Furthermore, the 1 PCA query needed for the reduction seems applicable regardless of whether the protocol uses IC [5], HIC [28], or directly m2F [here], to password-encrypt the pk. This stands in contention with the results in [5,28], where we believe the authors have missed this point. A guessing strategy does not work for indistinguishability-based definitions for the reasons discussed in “CPA versus iPCA” subsection. We also note that ANO-PCA was already used in the security proof from Pan and Zeng [27]. The authors claim that OCAKE protocol in [5] lacks *perfect forward secrecy* (PFS), but it is unclear whether the claim is attributed to the fact that the original proof for the protocol requires the underlying KEM to satisfy merely ANO-CPA.

6 Implementation and Performance Analysis

We make two preliminary notes on our instantiation of CHIC, which distinguish this work from previous proposals for building PAKE from a lattice-based KEM in the Ideal Cipher model.

Firstly, contrary to what has been suggested in previous papers [5,28], our security proof shows that the construction requires a KEM that offers more than just passive security (namely ANO-1PCA). For this reason, we take the (CCA-secure) Kyber standard defined in FIPS 203 [25] as the natural off-the-shelf lattice-based KEM instantiation. Indeed, Kyber has been shown to be IND-CCA [9,29] and ANO-CCA secure in [16,21,30].

Secondly, we recall that the bandwidth requirements of the IND-CCA version of Kyber are the same as that of the underlying IND-CPA construction: this is one of the properties of the Fujisaki-Okamoto transformation used by Kyber. For this reason, when it comes to bandwidth usage, our construction still outperforms previous proposals that (unjustifiably) propose to use the IND-CPA version. Indeed, there is no overhead in public-key transmission in the first flow of our protocol due to the compact half-ideal cipher, whereas in the second flow we have only the overhead of transmitting the (short) MAC tag.

It remains to show that the Kyber KEM satisfies the remaining requirements of having splittable and pseudo-uniform public keys. We refer to the full version of the paper [4] for a detailed proof that Kyber indeed satisfies the conditions outlined in Definition 4.

Our Implementation. We have implemented CHIC in C by extending the reference implementation of Kyber available from github.com/pqcrystals/kyber. The implementation is provided as supplementary material.

Before discussing parameter choices and giving some performance figures, we briefly describe how we implemented the three components of the compact

half-ideal cipher construction, as well as the computation of the MAC tag and key derivation hashes, which are all that's needed beyond Kyber KEM:

- **Ideal cipher over 256-bits:** We take the Rijndael variant that uses 256-bit blocks and 256-bit keys¹⁴. The code was taken from the open-source tool *ccrypt*, which in turn adapts on the original Rijndael reference implementation. We recall that this block cipher is used to hide the seed component ρ that results from public-key splitting.
- **Hashing to the Kyber polynomial ring R_q :** We reuse the implementation of the rejection sampling procedure that is used internally by Kyber to expand the public-key seed to a $k \times k$ matrix over R_q . The only difference to the Kyber implementation is that, rather than sampling a $k \times k$ matrix starting from a seed ρ , our implementation samples a vector of size k , seeded by the input to random oracle H in our m2F construction. We recall that the output of this procedure is used to mask the vector over R_q^k that results from public-key splitting using a group operation.
- **Masking vectors in R_q^k :** We reuse the functions already available in the Kyber code that permit adding and subtracting vectors over R_q^k .
- **Hashing to the key space of Rijndael:** We use SHA3-256 to produce the required 32-bytes.
- **Key Derivation Function and Tag Computation** Since these two hash functions take the same input, we implement them as a single SHA3-512 computation that produces 64 bytes, which we then split to obtain the session key (which is kept secret) and the tag (which is transmitted).

Parameter Selection. We consider all three variants of Kyber (Kyber512, Kyber768 and Kyber1024) as plausible instantiations for KEM. In our security analysis, we exclude the possibility of collisions on the block cipher, as well as on the hash function that derives the block cipher key. For this reason, we opt to use Rijndael with 256-bit block and key sizes. As a result, our parameter selection ensures at least 128-bit security against classical adversaries. We extend the discussion on the guarantees provided by CHIC in the conclusion section.

Performance Analysis. The bandwidth overhead of our protocol over Kyber KEM is minimal: it comprises of only 32-bytes for the tag in the second flow. Concerning execution time, Table 1 shows values in microseconds for the two stages of the initiator and the single stage of the responder for three cases: 1) using just the IND-CPA version of Kyber; 2) using just the IND-CCA version of Kyber; and 3) using CHIC. The measurements were taken in a modest laptop with a 2.3 GHz Intel “Core i5” processor with four cores, 128 MB of embedded eDRAM, a 6 MB shared level 3 cache, and 16 GB of RAM. We did not explore aggressive optimizations using parallelism (or even SIMD implementations), so these results can definitely be improved. The overhead in computation time for initiators is around 25% for Kyber 768 wrt the bare CCA KEM key exchange. For

¹⁴ Alternatively, we could employ standard AES and extend its domain from 128-bit to 256-bit blocks using the 3-round Feistel domain extender from [15], which is also indistinguishable from IC if AES itself is indistinguishable from IC.

responders, it is around 50%. Overall, these overheads decrease as the security level of Kyber increases, but the execution times are still in the order of tens of microseconds.

Table 1. Experimental results in microseconds. Comparison of execution times of CHIC participants (two initiator stages and responder single stage) with respect to key exchange using only a CPA or CCA Kyber KEM.

	CPA KEM			CCA KEM			CHIC		
	KeyGen	Enc	Dec	KeyGen	Enc	Dec	Start	Resp	End
Kyber512	25	29	9	45	49	12	70	74	14
Kyber768	28	36	41	49	59	65	75	85	93
Kyber1024	36	56	53	61	87	83	89	123	117

7 Conclusion

It’s important to underscore the significance of the 30+ years of research enabling CHIC’s development. The concept of constructing PAKE by encrypting a public key with a password dates back to EKE [7]. However, EKE and its variant OEKE [12] upon which EKE-KEM [28] and CHIC are based, were never standardized or deployed because of the difficulty of encrypting group elements with a password while preventing offline dictionary attacks. The work of [28] cleverly sidesteps the use of the IC over groups by using the m2F as a randomized cipher, which increases ciphertext size. CHIC avoids this ciphertext expansion by splitting Kyber pk and using the m2F instead as a keyed PRP. Our implementation represents the first practical realization of an approach previously deemed impractical [18], and it leverages the just-concluded ML-KEM standard [25].

This brings up the question: What security guarantees does CHIC provide against a quantum adversary? We proved that CHIC UC-realizes $\mathcal{F}_{\text{PAKE}}$ in the IC and RO model. The UC framework is the gold standard security definition for PAKE because it captures arbitrary password distributions. Unfortunately, it’s known that $\mathcal{F}_{\text{PAKE}}$ cannot be UC-realized without an idealized computation model or CRS [13]. In our implementation we instantiate the IC with Rijndael-256 and the RO with SHA3. These building blocks are quantum-resistant, but our security proof does not allow the adversary to query the RO and IC in superposition. Sufficiently large quantum computers don’t yet exist, but adversaries today can actively attack the protocol with classical computing capabilities and store PAKE transcripts now, hoping to decrypt them later with a quantum computer. However, such decryption would require breaking the KEM, and Kyber is quantum-resistant. This puts CHIC in the category of protocols secure against the “harvest-now/decrypt-later” quantum threat.

With a UC proof in the plain model aside, a consolidated analysis against adversaries that can actively attack the protocol with the support of a quantum

computer, demands working in the Quantum Random Oracle Model (QROM) [8] and/or Quantum Ideal Cipher Model (QICM) [19], which is a future direction to pursue. The QROM is better understood than the QICM [19]. To the best of our knowledge, the only PAKE protocols analysed in the UC framework and the QROM were only very recently proposed in [20]. While the authors' contribution is a significant step forward, a modular approach based on standard KEMs is likely yield a better candidate as a practical substitute of currently widely-used PAKE protocols. In this respect, CHIC is extremely efficient, with minimal overhead compared to the KEM it's based on.

Acknowledgements. We thank the anonymous reviewers of ASIACRYPT'24 for their valuable feedback. Afonso Arriaga and Marjan Škrobot received support from the Luxembourg National Research Fund (FNR) under the CORE Junior project (C21/IS/16236053/FuturePass).

References

1. Abdalla, M., Bellare, M., Neven, G.: Robust Encryption. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 480–497. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_28
2. Abdalla, M., Haase, B., Hesse, J.: Security analysis of CPace. In: Advances in Cryptology – ASIACRYPT 2021. pp. 711–741. Springer (2021)
3. Alnahawi, N., Hövelmanns, K., Hülsing, A., Ritsch, S.: Towards post-quantum secure PAKE - A tight security proof for OCAKE in the BPR model. In: Kohlweiss, M., Di Pietro, R., Beresford, A. (eds.) Cryptology and Network Security. CANS 2024. LNCS, vol. 14906, pp. 191–212. Springer, Singapore (2025). https://doi.org/10.1007/978-981-97-8016-7_9
4. Arriaga, A., Barbosa, M., Jarecki, S., Skrobot, M.: C'est très CHIC: A compact password-authenticated key exchange from lattice-based KEM. Cryptology ePrint Archive, Paper 2024/308 (2024), <https://eprint.iacr.org/2024/308>
5. Beguinet, H., Chevalier, C., Pointcheval, D., Ricosset, T., Rossi, M.: GeT a CAKE: Generic transformations from key encapsulation mechanisms to password authenticated key exchanges. In: Applied Cryptography and Network Security – ACNS 2023. pp. 516–538. Springer (2023)
6. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated Key Exchange Secure against Dictionary Attacks. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 139–155. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_11
7. Bellare, M., Merritt, M.: Encrypted key exchange: password-based protocols secure against dictionary attacks. In: Symposium on Research in Security and Privacy – S&P 1992. pp. 72–84. IEEE Computer Society (1992)
8. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random Oracles in a Quantum World. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25385-0_3
9. 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: European Symposium on Security and Privacy – EuroS&P 2018. pp. 353–367. IEEE Computer Society (2018)

10. Boyko, V., MacKenzie, P., Patel, S.: Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 156–171. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45539-6_12
11. Bradley, T., Camenisch, J., Jarecki, S., Lehmann, A., Neven, G., Xu, J.: Password-Authenticated Public-Key Encryption. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 2019. LNCS, vol. 11464, pp. 442–462. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21568-2_22
12. Bresson, E., Chevassut, O., Pointcheval, D.: Security proofs for an efficient password-based key exchange. In: ACM Conference on Computer and Communications Security – CCS 2003. pp. 241–250. Association for Computing Machinery (2003)
13. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.: Universally Composable Password-Based Key Exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_24
14. Coron, J.-S., Dodis, Y., Mandal, A., Seurin, Y.: A Domain Extender for the Ideal Cipher. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 273–289. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_17
15. Coron, J.-S., Dodis, Y., Mandal, A., Seurin, Y.: A Domain Extender for the Ideal Cipher. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 273–289. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_17
16. Grubbs, P., Maram, V., Paterson, K.G.: Anonymous, robust post-quantum public key encryption. In: Advances in Cryptology – EUROCRYPT 2022. pp. 402–432. Springer (2022)
17. Guo, C., Lin, D.: Improved domain extender for the ideal cipher. *Cryptography and Communications* **7**(4), 509–533 (2015). <https://doi.org/10.1007/s12095-015-0128-7>
18. Hao, F., van Oorschot, P.C.: SoK: Password-authenticated key exchange – theory, practice, standardization and real-world lessons. In: ACM Asia Conference on Computer and Communications Security – AsiaCCS 2022. pp. 697–711. Association for Computing Machinery (2022)
19. Hosoyamada, A., Yasuda, K.: Building Quantum-One-Way Functions from Block Ciphers: Davies-Meyer and Merkle-Damgård Constructions. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11272, pp. 275–304. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03326-2_10
20. Lyu, Y., Liu, S., Han, S.: Universal composable password authenticated key exchange for the post-quantum world. In: Advances in Cryptology – EUROCRYPT 2024. pp. 120–150. Springer (2024)
21. Maram, V., Xagawa, K.: Post-quantum anonymity of Kyber. In: Public-Key Cryptography – PKC 2023. pp. 3–35. Springer (2023)
22. Maurer, U., Renner, R., Holenstein, C.: Indifferentiability, Impossibility Results on Reductions, and Applications to the Random Oracle Methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24638-1_2
23. McQuoid, I., Rosulek, M., Roy, L.: Minimal symmetric PAKE and 1-out-of-n OT from programmable-once public functions. In: ACM Conference on Computer and Communications Security – CCS 2020. pp. 425–442. Association for Computing Machinery (2020)

24. Naehrig, M., Alkim, E., Bos, J., Ducas, L., Easterbrook, K., LaMacchia, B., Longa, P., Mironov, I., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D.: FrodoKEM. Tech. rep., National Institute of Standards and Technology (2020), available at <https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions>
25. NIST: FIPS203, Module-Lattice-based Key-Encapsulation Mechanism Standard. Federal Information Processing Standards Publication (2023), <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf>
26. Okamoto, T., Pointcheval, D.: REACT: Rapid enhanced-security asymmetric cryptosystem transform. In: Topics in Cryptology – CT-RSA 2001. pp. 159–174. Springer (2001)
27. Pan, J., Zeng, R.: A generic construction of tightly secure password-based authenticated key exchange. In: Advances in Cryptology – ASIACRYPT 2023. pp. 143–175. Springer (2023)
28. Santos, B.F.D., Gu, Y., Jarecki, S.: Randomized half-ideal cipher on groups with applications to UC (a)PAKE. In: Advances in Cryptology – EUROCRYPT 2023. pp. 128–156. Springer (2023)
29. Schwabe, P., Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Seiler, G., Stehlé, D., Ding, J.: CRYSTALS-KYBER. Tech. rep., National Institute of Standards and Technology (2022), available at <https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022>
30. Xagawa, K.: Anonymity of NIST PQC round 3 KEMs. In: Advances in Cryptology – EUROCRYPT 2022. pp. 551–581. Springer (2022)



Efficient Asymmetric PAKE Compiler from KEM and AE

You Lyu^{1,2}, Shengli Liu^{1,2}✉, and Shuai Han^{2,3}

¹ Department of Computer Science and Engineering, Shanghai Jiao Tong University,
Shanghai 200240, China

{[vergil](mailto:vergil@sjtu.edu.cn), [slliu](mailto:slliu@sjtu.edu.cn)}@sjtu.edu.cn

² State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China
dalen17@sjtu.edu.cn

³ School of Cyber Science and Engineering, Shanghai Jiao Tong University,
Shanghai 200240, China

Abstract. Password Authenticated Key Exchange (PAKE) allows two parties to establish a secure session key with a shared low-entropy password pw . Asymmetric PAKE (aPAKE) extends PAKE in the client-server setting, and the server only stores a password file instead of the plain password so as to provide additional security guarantee when the server is compromised.

In this paper, we propose a novel generic compiler from PAKE to aPAKE in the Universal Composable (UC) framework by making use of Key Encapsulation Mechanism (KEM) and Authenticated Encryption (AE).

- Our compiler admits efficient instantiations from lattice to yield lattice-based post-quantum secure aPAKE protocols. When instantiated with Kyber (the standardized KEM algorithm by the NIST), the performances of our compiler outperform other lattice-based compilers (Gentry et al. CRYPTO 2006) in all aspects, hence yielding the most efficient aPAKE compiler from lattice. In particular, when applying our compiler to the UC-secure PAKE schemes (Santos et al. EUROCRYPT 2023, Beguinet et al. ACNS 2023), we obtain the most efficient UC-secure aPAKE schemes from lattice.
- Moreover, the instantiation of our compiler from the tightly-secure matrix DDH (MDDH)-based KEM (Pan et al. CRYPTO 2023) can compile the tightly-secure PAKE scheme (Liu et al. PKC 2023) to a tightly-secure MDDH-based aPAKE, which serves as the first tightly UC-secure aPAKE scheme.

Keywords: Password authenticated key exchange · asymmetric password authenticated key exchange · lattice · post-quantum security

1 Introduction

Password Authenticated Key Exchange (PAKE) facilitates a secure establishment of session keys between two parties, say a client and a server, over a public network using a low entropy password pw . These session keys can then be used to set up secure communication channels for the client and server. In contrast

to Authenticated Key Exchange (AKE) protocols, PAKE operates with easily memorable passwords and eliminates the need for reliance on Public Key Infrastructure (PKI), hence offering a more convenient deployments of secure channels.

Note that the adversary can always implement online invocations of the PAKE protocol with guessing passwords. The low entropy of the passwords might make such online attacks succeed with non-negligible probability. So the security of PAKE requires these online attacks be the only efficient attacks.

For PAKE, a password must be shared between the client and server. Generally, the password is memorable so it is not necessary for the client to store it. However, the server has to store passwords for all the clients. If the server is compromised, the adversary can snatch the password pw from the server so as to impersonate the client. To address this vulnerability, asymmetric PAKE (aPAKE) [15] was proposed. With an aPAKE, the server only has to store a password file, which is a one-way function value $H(pw)$ of the password pw . Now if the server is compromised, the adversary only obtains the password file instead of the plain password. Therefore, aPAKE is deemed preferable to PAKE in the client-server setting. In this paper, we focus on aPAKE.

For aPAKE, if the adversary obtains $H(pw)$, it is possible for him/her to implement a so-called offline attack: compute or pre-compute $(pw', H(pw'))$ pairs to check the correctness of guessing password pw' by testing whether $H(pw') = H(pw)$. Clearly, offline attack is inevitable after $H(pw)$ is obtained by the adversary. As such, the security of aPAKE requires these offline attacks on $H(pw)$ be the only efficient attacks that may work (beyond the online attacks).

Security Models for aPAKE. There are two types of security notions for aPAKE, the game-based security in the Indistinguishability model (IND security) [3, 4] and the simulation-based security under the Universally Composable framework (UC security) [15]. The UC framework is preferable to the IND model in the following important aspects.

- The UC framework permits arbitrary correlations and distributions for passwords, while the IND model typically assumes passwords uniformly distributed over a dictionary for the sake of security proofs.
- Protocols that are proven secure in the UC framework enable a smooth and secure composition with other UC-secure protocols, thanks to the universal composition theorem [8].

Therefore, UC framework offers a more robust security guarantee and facilitates modular design of complex protocols, making it a better security model than the IND model.

aPAKE from Post-quantum Assumptions. Now there is a trend to migrate cryptographic algorithms to post-quantum ones. In fact, NIST has already determined Kyber [7] as the Post-Quantum Cryptography (PQC) winner for Key Encapsulation Mechanism (KEM), and Dilithium [11], Falcon [14], Sphincs⁺ [5] for digital signature schemes. Recently, the Crypto Forum Research Group

(CFRG) in IETF [32] initiated a selection process for both PAKE and aPAKE protocols. As far as we know, none of existing candidates are based on post-quantum assumptions.

There do exist a few aPAKE compilers instantiable from post-quantum assumptions. One compiler is the Ω -method proposed by Gentry et al. [15], which can compile any UC-secure PAKE protocol into a UC-secure aPAKE protocol with the help of a signature scheme. Accordingly, the existing UC-secure PAKE protocol from lattices [2, 30] and post-quantum secure signature schemes (like Dilithium [11], Falcon [14], Sphincs⁺ [5]) admit UC-secure aPAKE protocols. Another compiler is due to McQuoid and Xu [25], which can compile any UC-secure PAKE protocol into a strong aPAKE protocol. This strong aPAKE is secure against pre-computation attacks (in idealized generic group action model [12]), but their compiler can only be instantiable from group actions (e.g. isogeny-based CSIDH) rather than lattices. Note that there also exists an aPAKE compiler from UC-secure Key-Hiding AKE protocols [17, 30, 31]. However, no such key-hiding AKE protocols is instantiable from post-quantum assumptions. So up to now this compiler does not lead to post-quantum secure aPAKE yet.

In summary, the two UC-secure aPAKE compilers in [15, 25] seem to be the only available approaches to aPAKE achieving post-quantum security. However, either compiler has its own limitations.

- The compiler in [25] can only be instantiated from isogeny-based assumptions, rather than the mainstream lattice-based assumptions. Besides, the isogeny-based group action does not offer good computational efficiency under the current parameter configuration.
- The compiler in [15] can be instantiated from lattice-based assumptions with the help of lattice-based signature schemes. However, it is not very efficient since current lattice-based signature schemes (like Dilithium [11] in the NIST PQC winner) will lead to much higher computational overhead compared to their KEM counterparts (like Kyber [7] in the NIST PQC winner) under the same security level. Technically speaking, this might be mainly due to the time-consuming process of trapdoor sampling or rejection sampling involved in most lattice-based signature schemes, dating back to the seminal works of [16, 24].

This motivates us to seek a new aPAKE compiler instantiable from lattice to answer the following question:

*How to construct an efficient UC-secure aPAKE protocol
from lattices, without using signatures?*

Our Contribution. In this paper, we address the aforementioned question through the following contributions.

- **UC-secure aPAKE compiler without using signatures.** We propose a novel generic compiler that can compile any UC-secure PAKE protocol to a UC-secure aPAKE protocol, with the help of weak primitives of KEM

and Authenticated Encryption (AE). KEM is only required to have One-Wayness under Plaintext-Checkable Attacks (OW-PCA) [26], weaker than the standard Indistinguishability under Chosen-Ciphertext Attacks (IND-CCA). AE is only required to have one-time CCA security and one-time authenticity, which has efficient information-theoretical instantiations. In particular, our compiler does not rely on signature schemes.

- **Mutual explicit authentication and good round efficiency.** In addition to its excellent efficiency, our compiler enjoys other desirable features. By using our compiler, the resulting aPAKE enjoys mutual explicit authentication even if the underlying PAKE does not. This is similar to the compiler in [15] but in sharp contrast to [25], which completely relies on the underlying PAKE to achieve mutual explicit authentication.

Moreover, our compiler only augments two additional rounds to the underlying PAKE. If the last round message of PAKE is sent from the server, then the first additional round of our compiler can be merged to the last round of PAKE, and hence only one additional round is needed for aPAKE. This is also similar to the compiler in [15] and comparable to [25].¹

- **The most efficient UC-secure aPAKE from lattice.** By instantiating the KEM in our compiler with Kyber, the NIST PQC KEM winner, we obtain a lattice-based compiler. The performance evaluations show that our Kyber-based compiler outperforms the signature-based compiler in [15] from lattice in all aspects: ours save at least 61.1%-63.8% computing time and 41.5%-84.3% communication cost. Our Kyber-based compiler turns out to be the most efficient aPAKE compiler from lattice, and also results in *the most efficient UC-secure aPAKE schemes from lattice* when applying to the UC-secure PAKE schemes in [2, 30].
- **The first tightly UC-secure aPAKE.** By instantiating the KEM in our compiler with the tightly-secure matrix DDH (MDDH)-based KEM scheme proposed in [27], we obtain a tightness-preserving compiler, which yields *the first tightly UC-secure aPAKE scheme* when applying the compiler to the tightly-secure PAKE scheme in [22].

Technique Overview. Our aPAKE compiler mainly relies on a KEM with OW-PCA security to compile a PAKE to an aPAKE protocol. Here OW-PCA [26] is weaker than IND-CCA security and asks one-wayness of ciphertext even if the adversary has access to a “plaintext-checking” oracle which on input a ciphertext-key pair (c, K) determines whether c is an encapsulation of K .

Our compiler also requires a one-time secure Authenticated Encryption (AE) and five hash function H_0, \dots, H_4 . Here one-time security of AE includes one-time authenticity and one-time CCA security. We note that one-time secure AE has very efficient information-theoretical instantiations, e.g., from one-time pad and one-time secure message authentication code (MAC). On the other hand,

¹ More precisely, the compiler in [25] incurs an additional round from server to client before PAKE. Since it is usually the client who starts the protocol, actually two additional rounds are needed before PAKE in [25].

the hash functions will be modelled as Random Oracles (RO) during the security proof.

We refer to Fig. 1 for a high-level description of our compiler. More precisely, our compiler works as follows.

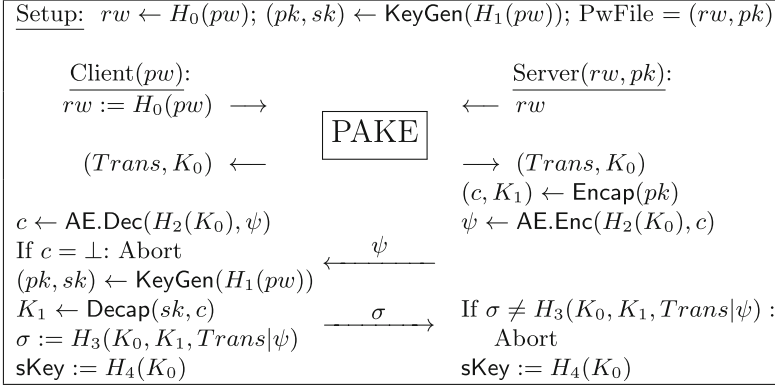


Fig. 1. High level description of our aPAKE compiler from KEM and AE.

The server's password file contains a hash value $H_0(pw)$ of the password pw and a KEM public key pk , which is derived from the password pw by using $H_1(pw)$ as the randomness of key generation, i.e., $(pk, sk) \leftarrow \text{KeyGen}(H_1(pw))$.

To establish a session key $sKey$, the client and server equip the underlying PAKE with password $rw = H_0(pw)$, and run the PAKE protocol to derive a PAKE session key K_0 .

Next, the server will use KEM to distribute a key K_1 to the client with the help of pk (retrieved from its password file): it first invokes $(c, K_1) \leftarrow \text{Encap}(pk)$ but it does not send c directly to the client. Instead, the server will use the PAKE session key K_0 to derive a key $H_2(K_0)$ for AE. Then with the help of AE, the server further encrypts the encapsulation c to obtain an AE ciphertext ψ via $\psi \leftarrow \text{AE.Enc}(H_2(K_0), c)$, and sends ψ to the client. In this way, the (one-time) authenticity of AE helps the server authenticate itself to the client since only the server and client can compute K_0 via PAKE and generate valid ciphertext ψ (i.e., the decryption of which does not lead to rejection) under AE key $H_2(K_0)$.

After receiving ψ , the client checks the validity of ψ with the AE key $H_2(K_0)$. If ψ is valid, i.e., $\text{AE.Dec}(H_2(K_0), \psi) \neq \perp$, the client will set the session key $sKey := H_4(K_0)$. It also recovers the KEM secret key sk from pw via $(pk, sk) \leftarrow \text{KeyGen}(H_1(pw))$, and then double decrypts ψ with AE key $H_2(K_0)$ and KEM secret key sk to obtain K_1 . Finally, the client computes an authenticator $\sigma := H_3(K_0, K_1, Trans|\psi)$ where $Trans$ is the transcription of PAKE, and sends σ to the server. Note that only the server and client can compute K_0 and K_1 . In this way, the confidentiality of KEM and AE (i.e., OW-PCA of KEM and one-time CCA of AE) help the client authenticate itself to the server.

Finally, the server checks the validity of σ with (K_0, K_1) by testing whether $\sigma = H_3(K_0, K_1, \text{Trans}|\psi)$, where $\text{Trans}, \psi, \sigma$ constitute the transcription seen by the server. If σ is valid, the server’s session key is set to $\text{sKey} := H_4(K_0)$.

Roughly speaking, our aPAKE compiler is designed in a “PAKE plus password-based mutual authentication” manner. PAKE helps the client and server share an AE key $H_2(K_0)$ and then AE helps the authentication of the server. Meanwhile, the server’s message ψ can be viewed as a “challenge” to the client. Only the client knowing the correct password pw is able to derive the KEM secret key sk to recover K_1 and generate the correct “response” σ with (K_0, K_1) . Putting differently, σ can be viewed as a proof of knowledge of pw for the client. As long as adversary \mathcal{A} does not obtain pw through offline or online attacks, \mathcal{A} cannot impersonate the client even if \mathcal{A} compromises the server and obtains (rw, pk) .

UC Security in RO Model. To prove the UC security for our aPAKE compiler in the random oracle (RO) model, we have to construct a simulator Sim to simulate all the interaction transcripts and session keys for both passive and active attacks implemented by \mathcal{A} , without the knowledge of password pw but accessing to the ideal functionality $\mathcal{F}_{\text{apake}}$ for aPAKE (see Fig. 4 in Sect. 2.3).

The aPAKE protocol has two parts: the PAKE part in which $rw = H_0(pw)$ is used to generate PAKE key K_0 , and the last two rounds in which K_0 and (pk, sk, rw) are used to generate ψ, σ and the session keys sKey . Recall that PAKE has UC security, so the PAKE part can emulate the ideal functionality $\mathcal{F}_{\text{pake}}$ (see Fig. 3 in Sect. 2.3 for the description of $\mathcal{F}_{\text{pake}}$).

Firstly, let us consider \mathcal{A} ’s attacks on the PAKE part. The task of Sim is to simulate the correctly distributed PAKE key K_0 without password pw . Note that if \mathcal{A} did not use the correct PAKE password $rw := H_0(pw)$ in the PAKE part, \mathcal{A} ’s active attack on the PAKE part hardly succeeds, due to the ideal functionality $\mathcal{F}_{\text{pake}}$ of PAKE. We consider the following three cases.

Case I: \mathcal{A} ’s successful active attacks on the PAKE part with the correct password pw . In this case, \mathcal{A} successfully implements either an online attack or an offline attack, in both of which \mathcal{A} must have queried $H_0(pw)$ to obtain rw . For \mathcal{A} ’s such an attack with rw , Sim can search the hash list to find pw' such that $rw = H_0(pw')$, and then justify the correctness of pw' with the help of Testpw interface of $\mathcal{F}_{\text{apake}}$. In this way, Sim is able to extract the correct password pw from \mathcal{A} ’s successful active attacks. Then with pw , it can simulate the PAKE part to generate PAKE key K_0 for \mathcal{A} .

Case II: \mathcal{A} ’s successful active attacks on the PAKE part with the stolen PAKE password rw . In this case, \mathcal{A} must have stolen the password file (rw, pk) and can use rw to impersonate client or server in the PAKE part of the aPAKE protocol. Sim can simulate the generation of password file for \mathcal{A} with random elements. More precisely, Sim picks rw and r randomly, sets $H_0(?) := rw$ and $H_1(?) := r$ respectively with $pw = ?$ undetermined, and generates $(pk, sk) \leftarrow \text{KeyGen}(r)$. If later \mathcal{A} attacks with correct pw , Sim can extract it (see Case I) and then re-program the hash function with

$H_0(pw) := rw$ and $H_1(pw) := r$. With the knowledge of rw , it can simulate \mathcal{F}_{pake} to generate PAKE key K_0 for \mathcal{A} . We stress that Sim generates (pk, sk, rw) without pw in this case.

Case III: the other cases. If neither Case I nor Case II happens, then either \mathcal{A} 's active attack fails or \mathcal{A} implements a passive attack.

Case III.1: Failed active attack on the PAKE part. In this case, either the PAKE part outputs \perp to abort the protocol (PAKE with explicit authentication), or the PAKE part outputs different PAKE session keys K_0 and K'_0 for Client and Server respectively (PAKE with implicit authentication), where K_0 and K'_0 must be uniform to \mathcal{A} according to the UC security of PAKE.

Case III.2: Passive attack on the PAKE part. In this case, the PAKE part generates the shared PAKE session key K_0 for Client and Server. According to the UC security, K_0 must be uniform to \mathcal{A} .

In this case, Sim can simulate K_0 (also K'_0) with a randomly chosen one.

Next, let us consider \mathcal{A} 's attacks on the last two rounds of the aPAKE protocol. We follow the above three cases.

Case I happened. In this case, Sim has extracted the correct password pw and also knows PAKE key K_0 . With the knowledge of pw and K_0 , Sim can give a perfect simulation for \mathcal{A} 's active attacks (here passive attack is impossible due to the previous active attack on the PAKE part) with round message ψ' or σ' . Note that Sim can use the correct pw to invoke \mathcal{F}_{apake} to assign the session key sKey determined by \mathcal{A} to both client and server.

Case II happened. In this case, Sim has the knowledge of (pk, sk, rw) itself (generated without pw) and the PAKE key K_0 , also uniformly distributed.

- For \mathcal{A} 's active attack with ψ' , Sim can determine the validity of ψ' with K_0 . It rejects ψ' and aborts the protocol for invalid ψ' , and for valid ψ' it invokes \mathcal{F}_{apake} to assign the session key sKey determined by \mathcal{A} to both client and server (dealing with the case that \mathcal{A} stole the file (pk, rw) and successfully impersonated server to accomplish the aPAKE protocol).
- For \mathcal{A} 's active attack with σ' , Sim can determine the validity of σ' with K_0 and K_1 . If σ' is valid and \mathcal{A} has obtained the correct password pw via an offline attack, it invokes \mathcal{F}_{apake} to assign the session key sKey determined by \mathcal{A} to both client and server. If σ' is invalid, it rejects σ' and aborts the protocol. Define Bad as the event that σ' is valid but \mathcal{A} did not obtain pw from offline attack. If Bad happens, it rejects σ' as well. However in the real experiment, in case of Bad , σ' should be accepted. This imperfect simulation does not annoy us since Bad can hardly happen, which is guaranteed by the OW-PCA security of KEM. The reason is as follows. Without pw , \mathcal{A} cannot derive sk , then \mathcal{A} cannot obtain K_1 due to the OW-PCA security, hence $H_3(K_0, K_1, \text{Trans}|\psi)$ is uniform to \mathcal{A} and $\sigma' = H_3(K_0, K_1, \text{Trans}|\psi)$ hardly holds.

Case III happened. In this case, Sim has the knowledge of K_0 which is uniformly distributed in \mathcal{A} 's view. Then to \mathcal{A} , the hash values of $H_2(K_0)$, $H_3(K_0, K_1, Trans|\psi)$, $H_4(K_0)$ are also uniformly distributed.

- For \mathcal{A} 's passive attacks on server or client, Sim can simulate ψ with $\psi \leftarrow \text{AE.Enc}(H_2(K_0), \text{dummy message})$ and simulate σ with a uniform string, and invoke \mathcal{F}_{apake} to generate random session key sKey for server or client. Given uniform AE key $H_2(K_0)$, AE encryption of a dummy message is indistinguishable from AE encryption of the real message c , due to the one-time CCA security of AE. Meanwhile random oracles guarantee the uniformity of $\sigma := H_3(K_0, K_1, Trans|\psi)$ and $\text{sKey} := H_4(K_0)$. Moreover, Sim invokes \mathcal{F}_{apake} to generate a uniform session key sKey for both client and server. Therefore, Sim's simulation is perfect for these passive attacks.
- For \mathcal{A} 's active attacks on client with ψ' , Sim directly rejects and aborts the protocol. Without the knowledge of AE key $H_2(K_0)$, \mathcal{A} 's forgery of ψ' will result in $\text{AE.Dec}(H_2(K_0), \psi') = \perp$, due to the one-time authenticity of AE. So Sim's simulation of dealing with ψ' is perfect (except with negligible probability).
- For \mathcal{A} 's active attacks on client with σ' , Sim directly rejects and aborts the protocol. Without the knowledge of KEM's secret key sk , K_0 , \mathcal{A} 's forgery of σ' can hardly collide with the valid $\sigma := H_3(K_0, K_1, Trans|\psi)$, due to uniformity of the RO outputs. So Sim's simulation of dealing with ψ' is perfect (except with negligible probability).

Comparison. In Table 1, we instantiate our aPAKE compiler with the NIST PQC KEM winner algorithm Kyber, and compare it with the signature-based compiler [15], which is the only known aPAKE compiler instantiable from lattices prior to our work. The performance results in Table 1 is obtained by running experiments on Intel(R) Core(TM) i7-1068NG7 CPU @ 2.30GHz with 4 cores under macOS 13.3.1. When the compiler [15] is instantiated with the NIST PQC lattice-based signature scheme Dilithium or Falcon, our compiler significantly outperforms theirs in all aspects including registration time, computing time, server storage size, and communication cost. This suggests that our compiler is the most efficient aPAKE compiler from lattices. As far as we know, there is no UC-secure aPAKE scheme constructed directly from lattices, and the only approach to UC-secure aPAKE from lattices is via compiling a PAKE scheme to an aPAKE. Therefore, the good efficiency of our compiler suggests that applying our compiler to the available UC-secure PAKE schemes from lattices will yield the most efficient UC-secure aPAKE schemes from lattices.

We also compare our compiler with known aPAKE compilers [15, 25] instantiated from other post-quantum assumptions in the full version of our paper [23]. The comparison shows that our compiler overwhelms these compilers in terms of registration time and computing time, and has comparable server storage and communication cost.

In Table 2, we also compare the resulting MDDH-based aPAKE scheme (from our MDDH-based compiler) with other aPAKE schemes in terms of tight UC

security. The tight security of aPAKE in [19, 22, 25, 31, 33] is eliminated by the impossibility result in [22], which essentially said that tightly UC-security is impossible to achieve if the secret derived from password by client is uniquely determined by the password file and moreover the relation between the secret and file can be efficiently determined. In contrast to their schemes, the client secret (rw, sk) in our MDDH-based aPAKE evades the impossibility result since the key generation of MDDH-based KEM makes sure that multiple sk correspond to one pk and hence to one password file. On the other hand, the aPAKE schemes [15, 17] in Table 2 are obtained by compiling PAKE + signature in [15] and AKE + ideal cipher in [17] respectively, but the security reduction is not tight, so it is unknown whether tight UC security can be achieved.

Table 1. Efficiency comparison of our Kyber-based compiler with the other aPAKE compilers [15] from lattices, where Kyber is in its version of “NIST security level 3”. For computing and communication cost, we use the mode of “**Client** + **Server** = **Total**” to reflect the separation of client/server computation/communication. The aPAKE compiler in [15] has two lattice-based instantiations, one is instantiated with Dilithium in its version of NIST security level 3, and the other is instantiated with Falcon in its version of NIST security level 1 (due to its lack of level 3).

aPAKE Compiler	Registration Time (ms)	Computing Time (ms)	Server Storage (KB)	Communication Cost (KB)	Assumption Type
[15] (Dilithium)	0.096	$0.245 + 0.126 = 0.371$	5.875	$3.215 + 3.938 = 7.153$	Lattice-based
[15] (Falcon)	7.003	$0.271 + 0.074 = 0.345$	2.189	$0.642 + 1.282 = 1.924$	Lattice-based
Ours (Kyber)	0.017	$0.051 + 0.083 = 0.134$	1.188	$0.031 + 1.094 = 1.125$	Lattice-based

Table 2. Comparison of our MDDH-based aPAKE scheme with other UC-secure aPAKE scheme in terms of tight UC security, where our aPAKE scheme is resulted from compiling the PAKE protocol [22] with our MDDH-based compiler. “?” denotes its tight UC security is unknown.

aPAKE scheme	Our MDDH-based aPAKE	[22]	[31]	[33]	[19]	[25]	[15]	[17]
Tight UC Security?	✓	×	×	×	×	×	?	?

The Necessity of AE in Our Compiler. Note that AE plays an important role in our compiler for aPAKE to achieve explicit authentication. Moreover, AE is especially necessary in our compiler when the underlying KEM has no anonymity under secret key leakage. No anonymity under secret key leakage means that given sk , one can efficiently decide whether a KEM ciphertext c is generated under pk . If we take off AE from the round message ψ , the server will send the KEM ciphertext c directly, and then the resulting aPAKE will suffer

from an offline attack: the adversary sees the encapsulation c , tries different password pw , generates $(pk, sk) \leftarrow \text{KeyGen}(H_1(pw))$, and tests whether c is the KEM ciphertext generated under key pair (pk, sk) .

As far as we know, many lattice-based KEMs have no anonymity under secret key leakage. Let us take Regev’s encryption scheme [29] as an example. For a ciphertext c generated under pk , the decryption of c using sk will result in a value which is either very close to 0 or close to $q/2$ with q the modulus. However, for c generated under another public key pk' , the decryption of c using sk may result in a value far from both 0 and $q/2$. In this way, one can in fact efficiently tell whether c is generated under pk (at least with noticeable probability). Consequently, AE is necessary for our lattice-based compiler.

On the other hand, if the underlying KEM does have anonymity under secret key, which means the adversary cannot distinguish a given ciphertext c is generated under (pk_1, sk_1) or (pk_2, sk_2) , the above offline attack fails. However, the price of getting rid of AE from our compiler is that we lose the explicit authentication of server-side. Actually, the CDH-based aPAKE compiler [19] can be seen as an instantiation of our compiler from ElGamal KEM without AE, and of course the CDH-based aPAKE does not achieve explicit mutual authentication.

Hardness on Achieving UC-Secure aPAKE in QROM. Note that our aPAKE compiler is proved under ROM instead of Quantum Random Oracle Model (QROM) [6]. QROM allows adversary \mathcal{A} to perform superposition queries to random oracles, making it hard for simulator to extract preimages or reprogram random oracles. It is especially hard to give UC-secure aPAKE protocols in QROM since reprogrammable random oracles are unavoidable for the construction of UC-secure aPAKE protocols, as demonstrated in [18]. Therefore, it is a big challenge to construct UC-secure aPAKE in QROM, and we leave it as an interesting open problem.

2 Preliminary

Let $\lambda \in \mathbb{N}$ denote the security parameter throughout the paper. If x is defined by y or the value of y is assigned to x , we write $x := y$. For $\mu \in \mathbb{N}$, define $[\mu] := \{1, 2, \dots, \mu\}$. Denote by $x \leftarrow \$ \mathcal{X}$ the procedure of sampling x from set \mathcal{X} uniformly at random. Let $|\mathcal{X}|$ denote the number of elements in \mathcal{X} . All our algorithms are probabilistic unless stated otherwise. We use $y \leftarrow \mathcal{A}(x)$ to define the random variable y obtained by executing algorithm \mathcal{A} on input x . We also use $y \leftarrow \mathcal{A}(x; r)$ to make explicit the random coins r used in the probabilistic computation. The notation “ \approx_s ” represents statistical indistinguishability, while “ \approx_c ” denotes computational indistinguishability. “PPT” abbreviates probabilistic polynomial-time. Denote by negl some negligible function.

In Subsect. 2.1, we recall the definitions of Key Encapsulation Mechanism (KEM), Authentication Encryption (AE) and their security notions. In Subsects. 2.2 and 2.3, we present the ideal functionalities of random oracle and (a)PAKE, respectively.

2.1 KEM and AE

Definition 1 (KEM). A key encapsulation mechanism (KEM) scheme $\text{KEM} = (\text{KeyGen}, \text{Encap}, \text{Decap})$ consists of three algorithms:

- $\text{KeyGen}(r)$: Taking as input a randomness $r \in \mathcal{R}$, the key generation algorithm outputs a pair of public key and secret key (pk, sk) .
- $\text{Encap}(pk)$: Taking as input a public key pk , the encapsulation algorithm outputs a pair of ciphertext $c \in \mathcal{CT}$ and encapsulated key $K \in \mathcal{K}$.
- $\text{Decap}(sk, c)$: Taking as input a secret key sk and a ciphertext c , the decapsulation algorithm outputs $K \in \mathcal{K}$.

The correctness of KEM requires that

$$\Pr \left[r \xleftarrow{\$} \mathcal{R}, (pk, sk) \leftarrow \text{KeyGen}(r), \right. \\ \left. (c, K) \leftarrow \text{Encap}(pk) : \text{Decap}(sk, c) = K \right] = 1 - \text{negl}(\lambda).$$

Definition 2 (OW-PCA security for KEM). For a KEM scheme $\text{KEM} = (\text{KeyGen}, \text{Encap}, \text{Decap})$, the advantage function of an adversary \mathcal{A} is defined by

$$\text{Adv}_{\text{KEM}}^{\text{ow-pca}}(\mathcal{A}) := \Pr \left[r \xleftarrow{\$} \mathcal{R}, (pk, sk) \leftarrow \text{KeyGen}(r), \right. \\ \left. (c, K) \leftarrow \text{Encap}(pk) : K = K' \right], \\ K' \leftarrow \mathcal{A}^{\text{CHECK}(sk, \cdot, \cdot)}(pk, c)$$

where the oracle CHECK takes as input a ciphertext-key pair (c, K) and returns whether $\text{Decap}(sk, c) = K$ or not. The OW-PCA security for KEM requires $\text{Adv}_{\text{KEM}}^{\text{ow-pca}}(\mathcal{A}) = \text{negl}(\lambda)$ for all PPT \mathcal{A} .

Definition 3 (AE). An authenticated encryption (AE) scheme $\text{AE} = (\text{Enc}, \text{Dec})$ consists of two algorithms:

- $\text{Enc}(k, m)$: Taking as input a key $k \in \mathcal{K}$ and a message $m \in \mathcal{M}$, the encryption algorithm outputs a ciphertext c .
- $\text{Dec}(k, c)$: Taking as input a key $k \in \mathcal{K}$ and a ciphertext c , the decryption algorithm outputs a message $m \in \mathcal{M}$ or a symbol \perp .

The correctness of AE requires that $\text{Dec}(k, \text{Enc}(k, m)) = m$ holds for all $k \in \mathcal{K}$ and all $m \in \mathcal{M}$.

Definition 4 (One-time authenticity for AE). For an AE scheme $\text{AE} = (\text{Enc}, \text{Dec})$, the advantage function of an adversary \mathcal{A} for one-time authenticity is defined by $\text{Adv}_{\text{AE}}^{\text{ot-auth}}(\mathcal{A}) := \Pr [k \xleftarrow{\$} \mathcal{K}, c \leftarrow \mathcal{A}^{\text{ENC}(k, \cdot)} : \text{Dec}(k, c) \neq \perp \wedge \text{cis not the output of oracle } \text{Enc}(k, \cdot)]$. The one-time authenticity for AE requires $\text{Adv}_{\text{AE}}^{\text{ot-auth}}(\mathcal{A}) = \text{negl}(\lambda)$ for all PPT \mathcal{A} that query the oracle $\text{ENC}(k, \cdot)$ at most once.

Definition 5 (One-time IND-CCA security for AE). For an AE scheme $\text{AE} = (\text{Enc}, \text{Dec})$, the advantage function of an adversary \mathcal{A} for one-time CCA security is defined by $\text{Adv}_{\text{AE}}^{\text{ot-cca}}(\mathcal{A}) := |\Pr [k \xleftarrow{\$} \mathcal{K}, b \xleftarrow{\$} \{0, 1\}, (m_0, m_1) \leftarrow \mathcal{A}^{\text{DEC}(k, \cdot)}, c_b \leftarrow \text{Enc}(k, m_b), b' \leftarrow \mathcal{A}^{\text{DEC}(k, \cdot)}(c_b) : b = b'] - 1/2|$. The one-time IND-CCA security requires $\text{Adv}_{\text{AE}}^{\text{ot-cca}}(\mathcal{A}) = \text{negl}(\lambda)$ for all PPT \mathcal{A} that query the oracle $\text{Dec}(k, \cdot)$ at most once for a ciphertext different from c_b .

Such an AE can be easily constructed from one-time pad and one-time secure Message Authentication Code (MAC), and hence it has a very efficient information-theoretical instantiation. See Sect. 4 for the detailed construction.

2.2 Idealized Random Oracle Model

The idealized functionality of random oracle is shown in Fig. 2. The idealized functionality of random oracle can be perfectly simulated by a PPT simulator Sim : Sim maintains a list \mathcal{L} (initialized to be empty) to store the records. Upon receiving a query (Eval, x) from P , Sim checks whether there exists $(x, y) \in \mathcal{L}$. If yes, Sim returns y . Otherwise, Sim randomly samples $y \leftarrow \mathcal{Y}$, records (x, y) in the list \mathcal{L} , and returns y .

Ideal Functionality \mathcal{F}_{RO}
The ideal functionality \mathcal{F}_{RO} is parameterized by a random map $H : \{0, 1\}^* \rightarrow \mathcal{Y}$.
Upon receiving a query (Eval, x) from P:
If $H(x)$ is defined: return $H(x)$.
Else: $y \leftarrow \mathcal{Y}$, $H(x) := y$, return $H(x)$.

Fig. 2. The ideal functionality \mathcal{F}_{RO} for random oracle H .

2.3 (Asymmetric) PAKE Under UC Framework

In Figs. 3 and 4, we show the ideal functionalities of PAKE and aPAKE respectively. Here the functionalities mainly follows from [15] along with some modifications as did in [33]. The ideal functionality of aPAKE is extended from that of PAKE, so we only explain aPAKE below.

Functionality \mathcal{F}_{pake}
The functionality \mathcal{F}_{pake} is parameterized by a security parameter λ . It interacts with an adversary \mathcal{A} and a set of parties via the following queries:
Upon receiving a query $(\text{NewClient}, C^{(i)}, iid, S^{(j)}, pw)$ from $C^{(i)}$:
Send $(\text{NewClient}, C^{(i)}, iid, S^{(j)})$ to \mathcal{A} . Record $(C^{(i)}, iid, S^{(j)}, pw)$ and mark it as fresh .
Upon receiving a query $(\text{NewServer}, S^{(j)}, iid', C^{(i)}, pw)$ from $S^{(j)}$:
Send $(\text{NewServer}, S^{(j)}, iid', C^{(i)})$ to \mathcal{A} . Record $(S^{(j)}, iid', C^{(i)}, pw)$ and mark it as fresh .
Upon receiving a query $(\text{Testpw}, P, iid, pw')$ from \mathcal{A}:
If there is a fresh record (P, iid, \cdot, pw) :
If $pw' = pw$, mark the record compromised and reply to \mathcal{A} with “correct guess”.
If $pw' \neq pw$, mark the record interrupted and reply \mathcal{A} with “wrong guess”.
Upon receiving a query $(\text{NewKey}, P, iid, sid, \text{Key}^*)$ from \mathcal{A}:
If there is a compromised record (P, iid, Q, pw) , send (sid, Key^*) to P .
If there is a fresh record (P, iid, Q, pw) , there is a completed record (Q, iid', P, pw') with $pw = pw'$, \mathcal{F}_{pake} has sent (sid, Key') to Q , and (Q, iid', P, pw') was fresh when (sid, Key') was sent, then output (sid, Key') to P .
In any other case, pick a new random key $\text{Key} \leftarrow \mathcal{K}$ and send (sid, Key) to P .
In all cases, mark the record (P, iid, Q, pw) as completed .

Fig. 3. The ideal functionality \mathcal{F}_{pake} for PAKE.

Functionality $\overline{\mathcal{F}}_{apake}$	
The functionality \mathcal{F}_{apake} is parameterized by a security parameter λ . It interacts with an adversary \mathcal{A} and a set of parties via the following queries:	
Password Storage	
Upon receiving a query (StorePWFile, $C^{(i)}, S^{(j)}, pw$) from $S^{(j)}$: Record (file, $C^{(i)}, S^{(j)}, pw$), mark it as fresh , and send (StorePWFile, $C^{(i)}, S^{(j)}$) to \mathcal{A} .	
Stealing Password File	
Upon receiving a query (StealPWFile, $C^{(i)}, S^{(j)}$) from \mathcal{A} : Mark record (file, $C^{(i)}, S^{(j)}, pw$) as compromised , and send (StealPWFile, $C^{(i)}, S^{(j)}$) to \mathcal{A} . If there is a record (offline, $C^{(i)}, S^{(j)}, pw'$) with $pw' = pw$: send pw and “correct guess” to \mathcal{A} .	
Upon receiving a query (OfflineTestPW, $C^{(i)}, S^{(j)}, pw'$) from \mathcal{A} : If there exists a record (file, $C^{(i)}, S^{(j)}, pw$) marked compromised : If $pw' = pw$, return “correct guess”; else return “wrong guess”. Else: Store (offline, $C^{(i)}, S^{(j)}, pw'$).	
Sessions	
Upon receiving a query (NewClient, $C^{(i)}, iid, S^{(j)}$) from $C^{(i)}$: Retrieve the record (file, $C^{(i)}, S^{(j)}, pw$). Send (NewClient, $C^{(i)}, iid, S^{(j)}$) to \mathcal{A} . Record ($C^{(i)}, iid, S^{(j)}, pw$) and mark it as fresh .	
Upon receiving a query (NewServer, $S^{(j)}, iid', C^{(i)}$) from $S^{(j)}$: Retrieve the record (file, $C^{(i)}, S^{(j)}, pw$). Send (NewServer, $S^{(j)}, iid', C^{(i)}$) to \mathcal{A} . Record ($S^{(j)}, iid, C^{(i)}, pw$) and mark it as fresh .	
Active Session Attacks	
Upon receiving a query (Testpw, P, iid, pw') from \mathcal{A} : If there is a fresh record (P, iid, \cdot, pw): If $pw' = pw$, mark the record compromised and reply to \mathcal{A} with “correct guess”. If $pw' \neq pw$, mark the record interrupted and reply to \mathcal{A} with “wrong guess”.	
Upon receiving a query (Impersonate, $C^{(i)}, iid$) from \mathcal{A} : If there exists a record (file, $C^{(i)}, S^{(j)}, pw$) marked compromised and a record ($C^{(i)}, iid, S^{(j)}, pw$) marked fresh : mark the record ($C^{(i)}, iid, S^{(j)}, pw$) compromised and return “correct guess” to \mathcal{A} . Otherwise, mark the record ($C^{(i)}, iid, S^{(j)}, pw$) interrupted and return “wrong guess” to \mathcal{A} .	
Key Generation	
Upon receiving a query (FreshKey, P, iid, sid) from \mathcal{A} : If there is a fresh record (P, iid, Q, pw) and sid has never been assigned to P 's any other instance: Pick $sKey \leftarrow \{0, 1\}^\lambda$, mark the record (P, iid, Q, pw) as completed , return ($sid, sKey$) to P , and record ($P, Q, sid, sKey$).	
Upon receiving a query (CopyKey, P, iid, sid) from \mathcal{A} : If there is a fresh record (P, iid, Q, pw), a completed record (Q, iid', P, pw) and sid has never been assigned to P 's any other instance: If there exists record ($Q, P, sid, sKey$) and the record (Q, iid', P, pw) was fresh when Q receives ($sid, sKey$): Mark the record (P, iid, Q, pw) as completed , and return ($sid, sKey$) to P .	
Upon receiving a query (CorruptKey, $P, iid, sid, sKey$) from \mathcal{A} : If there is a compromised record (P, iid, Q, pw) and sid has never been assigned to P 's any other instance: Mark the record (P, iid, Q, pw) as completed , and return ($sid, sKey$) to P .	
Upon receiving a query (Abort, P, iid) from \mathcal{A} : Mark the record (P, iid, \cdot, pw) as abort , and return (\perp, \perp) to P .	

Fig. 4. The ideal functionality \mathcal{F}_{apake} for aPAKE.

Security Guarantees in the Ideal World. \mathcal{F}_{apake} in Fig. 4 formalizes the ideal functionality of asymmetric PAKE with mutual explicit authentication. Roughly speaking, the ideal functionality \mathcal{F}_{apake} in the UC model captures the following security guarantees in the ideal world.

- **Passive attack and forward security.** For a passive attacker \mathcal{A} , the session key is uniformly distributed (which is modeled by the `FreshKey` and `CopyKey` query), even if the adversary is given the password.
- **Online guessing password attack.** In this case, the adversary \mathcal{A} guesses a password pw' once for a session and tests whether his guess is correct, which is modeled by the `Testpw` query. This means that any active attack implemented by a real-world adversary \mathcal{A} to the protocol can be translated to a single `Testpw` query in the ideal world. If the password-guess is correct, then the adversary can control the session key at its will (which is modeled by the `CorruptKey` query). Otherwise, the attacked party can detect the active attack and reject this session directly (which modeled by the `Abort` query).
- **Online impersonation attack on server after stealing the password file.** In this case, the adversary \mathcal{A} has obtained the password file stored in the server. Without loss of generality, we write the password file as $H(pw)$. With $H(pw)$, \mathcal{A} can impersonate a server perfectly to communicate with a client, which is modeled by the `Impersonate` query. Besides online attack, \mathcal{A} can also implement offline attacks by choosing a password pw' and checking whether $H(pw') = H(pw)$, which is modeled by the `OfflineTestPW` query. Moreover, before \mathcal{A} obtains the password file $H(pw)$, \mathcal{A} can precompute a table of possible pairs $(pw', H(pw'))$. When \mathcal{A} obtains the password file, he can search pw in the table quickly by comparing $H(pw') = H(pw)$. This is also captured by the `OfflineTestPW`. Note that in this case, \mathcal{A} cannot impersonate a client unless \mathcal{A} makes a correct online password guess, or \mathcal{A} obtains the password via offline attacks, i.e. \mathcal{A} gets a pw' s.t. $H(pw') = H(pw)$.

UC Security for aPAKE. In the real world, the environment \mathcal{Z} has all passwords for all users, controls the adversary \mathcal{A} , and sees all the interactions over the channel and session keys derived from the protocol. In UC framework, we will construct a PPT simulator `Sim` which has access to the ideal functionality \mathcal{F}_{apake} and interacts with the environment \mathcal{Z} . If the view simulated by `Sim` for environment \mathcal{Z} is indistinguishable to \mathcal{Z} 's view in the real world, then UC security is achieved.

Difference Between Our $\mathcal{F}_{pake}/\mathcal{F}_{apake}$ and the Original $\mathcal{F}_{pake}/\mathcal{F}_{apake}$. The original ideal functionality \mathcal{F}_{pake} for PAKE was proposed in [9]. The ideal functionality \mathcal{F}_{apake} for aPAKE in [15] was built upon [9]. Shoup [33] improved and optimized both \mathcal{F}_{pake} and \mathcal{F}_{apake} in several ways.

- In the original ideal functionality \mathcal{F}_{pake} and \mathcal{F}_{apake} , each protocol instance must be identified by a globally unique session identifier and the PAKE participants can successfully establish a joint session key only if they use the same session identifier sid . The requirements on sid are problematic for an implementer, as pointed out by [1]. Shoup's modified \mathcal{F}_{pake} addresses this issue by replacing globally unique session identifiers with locally unique instance identifiers (iid), and session identifiers (sid) are seen as protocol outputs.
- The \mathcal{F}_{apake} in [15] is flawed, as pointed out by [18]. In fact, Shoup also pointed it out, and he fixed these flaws with a modified \mathcal{F}_{apake} .

- \mathcal{F}_{apake} in [33] models the explicit authentication (via fresh-key, copy-key, corrupt-key and abort interfaces), while the formulation of explicit authentication in [15] is flawed, as pointed out by Remark 10 in [33].

In our paper, we adopt the optimized \mathcal{F}_{pake} and \mathcal{F}_{apake} provided in [33].

Remark 1. In this paper, we assume that each instance of a party (client or server) is initialized with the different instance ID *iid*. This is reasonable and also taken in [33]. Another notation is that for simplicity, our ideal functionality \mathcal{F}_{apake} does not model the case that client mistypes its password, but this can be easily taken into account with the same approach in [33] (See Remark 3 and Remark 4 in [33] for more details).

3 Our aPAKE Compiler from KEM and AE

In this section, we propose an aPAKE compiler from KEM and AE, and show how to construct aPAKE from UC-secure PAKE with the help of our compiler in the UC framework. The detailed compiler (as well as the aPAKE construction) is shown in Fig. 5. Due to its UC security, PAKE can emulate its ideal functionality \mathcal{F}_{pake} , so we replace PAKE with \mathcal{F}_{pake} in the aPAKE construction.

Clearly, the resulting aPAKE scheme from our compiler is correct if the underlying PAKE, KEM and AE scheme are correct. If the underlying KEM has OW-PCA security and AE has one-time authenticity and one-time CCA security, then our compiler is able to compile a UC-secure PAKE to a UC-secure aPAKE, as shown in the following theorem.

Theorem 1. *If KEM is a key encapsulation mechanism with OW-PCA security, AE is an authenticated encryption scheme with one-time authenticity and one-time CCA security, H_0, H_1, H_2, H_3, H_4 work as random oracles, then the aPAKE scheme in Fig. 5 securely emulates \mathcal{F}_{apake} , hence achieving UC security in the $\{\mathcal{F}_{pake}, \mathcal{F}_{RO}\}$ -hybrid model. More precisely, suppose there are at most N parties, ℓ sessions and q random oracle queries, then there exists a simulator Sim s.t.*

$$\begin{aligned} |\Pr[\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}] - \Pr[\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}}]| \leq & N^2 \ell \cdot \text{Adv}_{\text{KEM}}^{\text{ow-pca}}(\mathcal{B}_{\text{KEM}}) + \frac{q^2 + q + 1}{2^\lambda} \\ & + \ell \cdot \text{Adv}_{\text{AE}}^{\text{ot-auth}}(\mathcal{B}_{\text{AE}}) + \ell \cdot \text{Adv}_{\text{AE}}^{\text{ot-cca}}(\mathcal{B}_{\text{AE}}). \end{aligned}$$

PROOF. The main objective of the proof is constructing a PPT simulator Sim to simulate an indistinguishable view for the environment \mathcal{Z} . Sim is designed to have access to the ideal functionality \mathcal{F}_{apake} and interact with the environment \mathcal{Z} , thereby emulating the real-world aPAKE protocol interactions involving the adversary \mathcal{A} , the parties, and the environment \mathcal{Z} . It is important to note that Sim *does not possess any password*.

Let $\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}$ represent the real-world experiment where the environment \mathcal{Z} interacts with the parties and adversary \mathcal{A} who can access ideal functionality \mathcal{F}_{pake} via Testpw and NewKey. Let $\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}}$ represents the ideal experiment

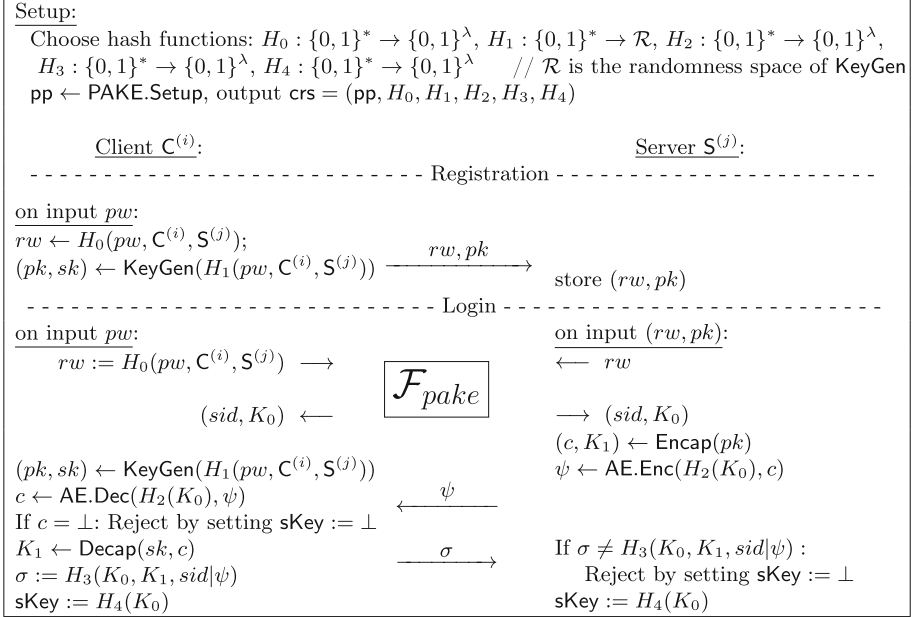


Fig. 5. Construction of UC-secure aPAKE from UC-secure PAKE with our compiler.

where \mathcal{Z} interacts with “dummy” parties and simulator Sim . By $\mathbf{Real}_{\mathcal{Z}, \mathcal{A}} \Rightarrow 1$, we mean \mathcal{Z} outputs 1 in $\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}$, and $\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}} \Rightarrow 1$ is similarly defined.

Our goal is to show that $|\Pr[\mathbf{Real}_{\mathcal{Z}, \mathcal{A}} \Rightarrow 1] - \Pr[\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}} \Rightarrow 1]|$ is negligible by employing a series of games, denoted as Game \mathbf{G}_0 – \mathbf{G}_7 . In this sequence, \mathbf{G}_0 corresponds to $\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}$, while \mathbf{G}_7 corresponds to $\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}}$. We aim to show that these adjacent games are indistinguishable from the view of \mathcal{Z} . For simplicity, we write $H_0(pw, \mathcal{C}^{(i)}, \mathcal{S}^{(j)})$ and $H_1(pw, \mathcal{C}^{(i)}, \mathcal{S}^{(j)})$ as $H_0(pw)$ and $H_1(pw)$ in the security proof.

Game \mathbf{G}_0 . This is the real experiment $\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}$. In this experiment, \mathcal{Z} initializes a password for each client-server pair, sees the interactions among clients, servers, ideal functionality \mathcal{F}_{pake} and adversary \mathcal{A} , and also obtains the corresponding session keys of protocol instances. During the execution, \mathcal{F}_{pake} may be invoked to create records like “ (P, iid, Q, rw) ” which we call *inner records* so as to distinguish them from the records created by \mathcal{F}_{apake} . Here \mathcal{A} may implement attacks like view, modify, insert, or drop messages over the network. In \mathbf{G}_0 , H_0, H_1, H_2, H_3, H_4 works as random oracles. Each party will do the following.

- For a server $\mathcal{S}^{(j)}$ on input $(\text{StorePWFile}, \mathcal{C}^{(i)}, \mathcal{S}^{(j)}, pw)$ from \mathcal{Z} , it computes $rw := H_0(pw)$ and $(pk, sk) \leftarrow \text{KeyGen}(H_1(pw))$ and then stores $(\mathcal{C}^{(i)}, \mathcal{S}^{(j)}, rw, pk)$ locally.
- For a server $\mathcal{S}^{(j)}$ on input $(\text{StealPWFile}, \mathcal{C}^{(i)}, \mathcal{S}^{(j)})$ from \mathcal{A} , it retrieves $(\mathcal{C}^{(i)}, \mathcal{S}^{(j)}, rw, pk)$ from local storage, and returns (rw, pk) to \mathcal{A} .

- For a client instance $(C^{(i)}, iid)$ on input $(\text{NewClient}, C^{(i)}, iid, S^{(j)}, pw)$ from \mathcal{Z} , it computes $rw := H_0(pw)$ and issues “ $(\text{NewClient}, C^{(i)}, iid, S^{(j)}, rw)$ ” query to ideal functionality $\mathcal{F}_{\text{pake}}$. According to the specification of $\mathcal{F}_{\text{pake}}$, it will create a fresh inner record “ $(\text{NewClient}, C^{(i)}, iid, S^{(j)}, rw)$ ”, and send “ $(\text{NewClient}, C^{(i)}, iid, S^{(j)})$ ” to \mathcal{A} .
If \mathcal{A} issues “ $(\text{NewKey}, C^{(i)}, iid, sid, \text{Key}^*)$ ” to $\mathcal{F}_{\text{pake}}$, then the instance $(C^{(i)}, iid)$ may receive “ $(sid, \text{Key} = K_0)$ ” from $\mathcal{F}_{\text{pake}}$.
- For a server instance $(S^{(j)}, iid')$ on input $(\text{NewServer}, S^{(j)}, iid', C^{(i)})$ from \mathcal{Z} , it retrieves $(C^{(i)}, S^{(j)}, rw, pk)$ from its storage, and sends “ $(\text{NewServer}, S^{(j)}, iid', C^{(i)}, rw)$ ” to $\mathcal{F}_{\text{pake}}$. According to the specification of $\mathcal{F}_{\text{pake}}$, it will create a fresh inner record “ $(S^{(j)}, iid', C^{(i)}, rw)$ ”, and send “ $(\text{NewServer}, S^{(j)}, iid', C^{(i)})$ ” to \mathcal{A} .
If \mathcal{A} issues “ $(\text{NewKey}, S^{(j)}, iid', sid, \text{Key}^*)$ ” to $\mathcal{F}_{\text{pake}}$, then the instance $(S^{(j)}, iid')$ may receive “ $(sid, \text{Key} = K_0)$ ” from $\mathcal{F}_{\text{pake}}$.
- \mathcal{A} can access $\mathcal{F}_{\text{pake}}$ via two interfaces $(\text{Testpw}, P, iid, pw)$ and $(\text{NewKey}, P, iid, sid, \text{Key}^*)$.
 - Upon $\mathcal{F}_{\text{pake}}$ receiving “ $(\text{NewKey}, P, iid, sid, \text{Key}^*)$ ” from \mathcal{A} , $\mathcal{F}_{\text{pake}}$ may create “ $(sid, \text{Key} = K_0)$ ” according to the specification in Fig. 3, mark the inner record “ (P, iid, Q, rw) ” as completed, and send “ $(sid, \text{Key} = K_0)$ ” to the instance “ (P, iid) ”.
 - Upon $\mathcal{F}_{\text{pake}}$ receiving “ $(\text{Testpw}, P, iid, rw')$ ” query from \mathcal{A} , it checks whether $rw' = H_0(pw)$. If yes, $\mathcal{F}_{\text{pake}}$ returns “correct guess” to \mathcal{A} . Otherwise, $\mathcal{F}_{\text{pake}}$ returns “wrong guess” to \mathcal{A} . Meanwhile, $\mathcal{F}_{\text{pake}}$ marks the inner record “ (P, iid, Q, rw) ” as compromised or interrupted accordingly (See Fig. 3).
- Upon a server instance $(S^{(j)}, iid')$ receiving message (sid, K_0) from $\mathcal{F}_{\text{pake}}$, it retrieves “ $(C^{(i)}, S^{(j)}, rw, pk)$ ” from its storage, and computes $(c, K_1) \leftarrow \text{Encap}(pk)$. Then it computes $\psi \leftarrow \text{AE.Enc}(H_2(K_0), c)$ and sends ψ to \mathcal{A} .
- Upon a client instance $(C^{(i)}, iid)$ receiving message “ (sid, K_0) ” from $\mathcal{F}_{\text{pake}}$ and message ψ from \mathcal{A} , it first decrypts $c \leftarrow \text{AE.Dec}(H_2(K_0), \psi)$. If $c = \perp$, then it rejects and sends \perp to \mathcal{Z} . Otherwise, it computes $(pk, sk) \leftarrow \text{KeyGen}(H_1(pw))$ and decrypts $K_1 \leftarrow \text{Decap}(sk, c)$. Finally, it computes $\sigma := H_3(K_0, K_1, sid|\psi)$, sends σ to \mathcal{A} , sets $\text{sKey} := H_4(K_0)$ and returns sKey to \mathcal{Z} .
- Upon a server instance $(S^{(j)}, iid')$ receiving message σ from \mathcal{A} , if $\sigma \neq H_3(K_0, K_1, sid|\psi)$, then it rejects and sends \perp to \mathcal{Z} . Otherwise, it sets $\text{sKey} := H_4(K_0)$ and returns sKey to \mathcal{Z} .

We have

$$\Pr[\mathbf{Real}]_{\mathcal{Z}, \mathcal{A}} \Rightarrow 1 = \Pr[\mathbf{G}_0 \Rightarrow 1].$$

Game \mathbf{G}_1 (Simulations for Clients and Servers with pw). In this game, we introduce a simulator Sim who *additional knows passwords* and has access to the ideal functionality $\mathcal{F}_{\text{apake}}$. Now in Game \mathbf{G}_1 , the client and server become “dummy party” and directly forward their inputs to the ideal functionality $\mathcal{F}_{\text{apake}}$ defined in Fig. 4. Then Sim simulates the behaviors of clients and servers with the help of pw as follows.

- For a dummy server $S^{(j)}$ on input $(\text{StorePWFile}, C^{(i)}, S^{(j)}, pw)$ from \mathcal{Z} , it directly sends this query to \mathcal{F}_{apake} . Then \mathcal{F}_{apake} sends $(\text{StorePWFile}, C^{(i)}, S^{(j)})$ to Sim . Then Sim simulates the password file (rw, pk) with $rw := H_0(pw)$, $(pk, sk) \leftarrow \text{KeyGen}(H_1(pw))$. Sim also stores a trapdoor record $(C^{(i)}, S^{(j)}, rw, pk, sk, H_1(pw))$ in its local storage.
- For a dummy server $S^{(j)}$ on input $(\text{StealPWFile}, C^{(i)}, S^{(j)})$ from \mathcal{A} , Sim directly sends this query to \mathcal{F}_{apake} . Then \mathcal{F}_{apake} functions as described in Fig. 4 and sends $(\text{StealPWFile}, C^{(i)}, S^{(j)})$ to Sim . Then Sim returns the password file (rw, pk) to \mathcal{A} .
- For a dummy client instance $(C^{(i)}, iid)$ on input $(\text{NewClient}, C^{(i)}, iid, S^{(j)}, pw)$, it directly sends this query to \mathcal{F}_{apake} . Then \mathcal{F}_{apake} sends $(\text{NewClient}, C^{(i)}, iid, S^{(j)})$ to Sim and Sim simulates the behavior of the client instance $(C^{(i)}, iid)$ with the password pw , just like G_0 .
- For a dummy server instance $(S^{(j)}, iid')$ on input $(\text{NewServer}, S^{(j)}, iid, C^{(i)})$, it directly sends this query to \mathcal{F}_{apake} . Then \mathcal{F}_{apake} sends $(\text{NewServer}, S^{(j)}, iid', C^{(i)})$ to Sim and Sim simulates the behavior of the server instance $(S^{(j)}, iid')$ with the PAKE password rw , just like G_0 .
- For dummy client and server instances, the generations of ψ , σ and $s\text{Key}$ are all simulated by Sim with the password pw , just like G_0 .

With the knowledge of *passwords*, the simulations of the behaviors of all clients and servers are perfect.

Moreover, Sim also simulates random oracles H_i ($i \in [0, 4]$) maintaining separate lists, namely \mathcal{L}_{H_i} . For a query x on $H_i(\cdot)$, if $(x, y) \in \mathcal{L}_{H_i}$, then Sim will return y as the reply. Otherwise, Sim will choose a random element y , record (x, y) in \mathcal{L}_{H_i} , and return y as the reply.

During the simulation, Sim additionally checks a bad event: if there exists two different random oracle queries to H_i such that $H_i(pw) = H_i(pw')$, then Sim will abort the game. By the ideal functionality of random oracles, Sim 's simulations for oracles H_i are perfect except a collision occurs in the simulation of H_i . Suppose that the adversary issues q random oracle queries totally, by the union bound, we have

$$|\Pr[G_1 \Rightarrow 1] - \Pr[G_0 \Rightarrow 1]| \leq \frac{q^2}{2\lambda}.$$

The following games will change the simulations of Sim step by step in an indistinguishable way so that Sim can arrive at its final form, and accomplish the simulations in $\text{Ideal}_{\mathcal{Z}, \text{Sim}}$ without passwords pw . The complete description of the simulator Sim is given in the full version [23].

Game G_2 (Simulation for Ideal Functionality \mathcal{F}_{pake} Without pw). In G_2 , simulator Sim will simulate the ideal functionality \mathcal{F}_{pake} itself, but without the knowledge of pw . More precisely, Sim will maintain some inner records to simulate the output of \mathcal{F}_{pake} in the following way.

- **Upon receiving** $(\text{NewClient}, C^{(i)}, iid, S^{(j)})$ **from** \mathcal{F}_{apake} : Sim sends $(\text{NewClient}, C^{(i)}, iid, S^{(j)})$ to \mathcal{A} . Then it stores “ $(C^{(i)}, iid, S^{(j)}, ?)$ ” as an inner record and marks it as **fresh**.

- **Upon receiving input** $(\text{NewServer}, S^{(j)}, iid', C^{(i)})$ **from** \mathcal{F}_{apake} : Sim sends $(\text{NewServer}, S^{(j)}, iid', C^{(i)})$ to \mathcal{A} . Then it stores “ $(S^{(j)}, iid', C^{(i)}, ?)$ ” as an inner record and marks it as **fresh**.
 - **Upon receiving a query** $(\text{Testpw}, P, iid, rw)$ **from** \mathcal{A} : If there exists a **fresh** inner record “ $(P, iid, Q, ?)$ ”, then do the following.
 1. If there exists $(pw, rw) \in \mathcal{L}_{H_0}$, then Sim sends $(\text{Testpw}, P, iid, pw)$ to \mathcal{F}_{apake} and forwards \mathcal{F}_{apake} ’s reply (“correct guess” or “wrong guess”) to \mathcal{A} .
 2. If \mathcal{A} ever issued $(\text{StealPWFile}, C^{(i)}, S^{(j)})$ query before and Sim returned (rw', pk') to \mathcal{A} with $rw' = rw$, and (P, iid) is an instance among the interaction of $C^{(i)}$ and $S^{(j)}$, then Sim returns “correct guess” to \mathcal{A} .
 3. In other cases, return “wrong guess”.
 4. If Sim returns “correct guess” to \mathcal{A} , it also replaces “ $(P, iid, Q, ?)$ ” with “ (P, iid, Q, rw) ” and marks it as a **compromised** inner record. Otherwise, it marks “ $(P, iid, Q, ?)$ ” as an **interrupted** inner record.
 - **Upon receiving a query** $(\text{NewKey}, P, iid, sid, \text{Key}^*)$ **from** \mathcal{A} : The simulator replies the query just like \mathcal{F}_{pake} .
 1. If sid has been assigned to P ’s any other instance (P, iid') , return \perp .
 2. If there exists a **compromised** inner record “ (P, iid, Q, rw) ”, then output (sid, Key^*) to P .
 3. If there exists a **fresh** inner record “ $(P, iid, Q, ?)$ ” and a **completed** inner record “ $(Q, iid', P, ?)$ ”, “ (sid, Key') ” was sent to Q and $(Q, iid', P, ?)$ was **fresh** at the time, then output “ (sid, Key') ” to P .
 4. In any other case, pick a new random key $\text{Key} \leftarrow \mathcal{K}$ and send “ (sid, Key) ” to P .
- Finally, mark the inner record “ (P, iid, Q, \cdot) ” as **completed**.

It is easy to see the simulation of $\text{NewClient}, \text{NewServer}, \text{NewKey}$ query is perfect. The only difference in G_1 and G_2 occurs in the simulation of Testpw query.

Note that in G_1 , $(\text{Testpw}, P, iid, rw)$ returns “correct guess” if and only if rw is the PAKE password used in the instance (P, iid) , which is the case that $rw = H_0(pw)$ for the input pw to party P . Therefore, G_2 and G_1 differs only when \mathcal{A} issues a $(\text{Testpw}, P, iid, rw)$ query to \mathcal{F}_{pake} , $rw = H_0(pw)$ but \mathcal{A} does not query $H_0(pw)$. By the ideal functionality of random oracle, $H_0(pw)$ is uniformly distributed to \mathcal{A} if \mathcal{A} does not query it. So we have

$$|\Pr[G_2 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \frac{1}{2^\lambda}.$$

Game G_3 (Replace Hash Values of K_0 with Uniform Ones Unless \mathcal{A} Implements a Successful On-Line Active Attacks on \mathcal{F}_{pake}). We consider the following three cases, where Case I and Case II cover \mathcal{A} ’s successful on-line active attacks on \mathcal{F}_{pake} , and Case III covers the rest.

Case I. \mathcal{A} implements a (successful) on-line attack on \mathcal{F}_{pake} with the correct password pw . That is, \mathcal{A} issues “ $(\text{Testpw}, P, iid, rw = H(pw))$ ” query, and then Sim obtains “correct guess” from \mathcal{F}_{apake} for its query “ $(\text{Testpw}, P, iid, pw)$ ” to \mathcal{F}_{apake} . In this case, Sim is able to extract the correct password pw .

Case II. \mathcal{A} implements a (successful) on-line attack on \mathcal{F}_{pake} by impersonating a party with the stolen PAKE password rw . That is \mathcal{A} first issues a query (StealPWFFile, P, Q), then issues “(Testpw, P, iid, rw)” or “(Testpw, Q, iid, rw)” query. In this case, \mathcal{A} has stolen the PAKE password rw and successfully implements an on-line impersonation attack.

Case III. Neither Case I nor Case II occurs.

If Case I or Case II happens, G_3 is the same as G_2 . But if Case III happens in G_3 , when \mathcal{A} issues a “(NewKey, P, iid, Key^*)” query resulting in K_0 , then the hash values of $H_2(K_0)$, $H_3(K_0, K_1, c)$ and $H_4(K_0)$ are replaced with independently and uniformly chosen elements, no matter whether \mathcal{A} has ever queried any of them or not.

It is easy to see that G_3 is the same as G_2 unless \mathcal{A} has ever queried $H_2(K_0)$, $H_3(K_0, K_1, c)$ or $H_4(K_0)$ in Case III.

Note that Case III means that either there is no on-line attacks from \mathcal{A} or the on-line attack does not succeed. Upon \mathcal{A} issuing a “(NewKey, P, iid, Key^*)” query to \mathcal{F}_{pake} in Case III, then the resulting key K_0 simulated by Sim is uniformly distributed to \mathcal{A} (according to the specification of the simulation of ideal functionality \mathcal{F}_{pake} , the session key K_0 should be uniform). Suppose the adversary totally issues q random oracle queries, then \mathcal{A} ever issues hash query $H_2(K_0)$, $H_3(K_0, K_1, c)$ or $H_4(K_0)$ with correct K_0 with probability at most $q/2^\lambda$. So we have

$$|\Pr [G_3 \Rightarrow 1] - \Pr [G_2 \Rightarrow 1]| \leq \frac{q}{2^\lambda}.$$

Now in Case III, the AE key $H_2(K_0)$, the authenticator $\sigma = H_3(K_0, K_1, c)$ and the session key $sKey = H_4(K_0)$ will be uniformly distributed to \mathcal{A} . Jumping ahead, the uniform AE key $H_2(K_0)$ in Case III paves the way for the security reduction to the security of AE in G_4 and G_5 .

Now in G_3 , Sim does not need pw to simulate the generation of PAKE session key K_0 : upon \mathcal{A} 's (NewKey, P, iid, Key^*) query, Sim just sets $K_0 := Key^*$ in Case I and Case II, and can choose K_0 uniformly in Case III. But Sim still needs rw to identify Case II and needs password pw to generate password file. Moreover, the generations of ψ and σ also needs pw : $(pk, sk) \leftarrow \text{KeyGen}(H_1(pw))$, $(c, K_1) \leftarrow \text{Encap}(pk)$, $\psi \leftarrow \text{AE.Enc}(H_2(K_0), c)$, and $\sigma = H_3(K_0, K_1 = \text{Decap}(sk, c), \text{sid}|\psi)$.

Game G_4 (Simulation of Generating Server's Message ψ Without pw).

In G_4 , Sim is the same as in G_3 , except for Sim's generation of ψ for the server instance $(S^{(j)}, iid')$. We describe Sim's simulations in the three cases (which are defined in G_3) in G_4 .

- If Case I occurs to $(S^{(j)}, iid')$, then Sim does not need to know pw beforehand. Instead, Sim can extract the true password pw of server $S^{(j)}$ from \mathcal{F}_{apake} . Then Sim can use pw to generate $(pk, sk) \leftarrow \text{KeyGen}(H_1(pw))$, $(c, K_1) \leftarrow \text{Encap}(pk)$ and $\psi \leftarrow \text{AE.Enc}(H_2(K_0), c)$, exactly like G_3 .
- If Case II occurs to $(S^{(j)}, iid')$, then Sim does not need the knowledge of pw to generate ψ . In this case, Sim directly retrieves pk from its trapdoor record $(C^{(i)}, S^{(j)}, rw, pk, sk, r)$, and then compute $(c, K_1) \leftarrow \text{Encap}(pk)$ and generates $\psi \leftarrow \text{AE.Enc}(H_2(K_0), c)$. Simulation of ψ is exactly the same as G_3 .

- If neither Case I nor Case II occurs to $(S^{(j)}, iid')$, then Sim does not need the knowledge of pw to generate ψ either. In this case, Sim directly retrieves pk from its trapdoor record $(C^{(i)}, S^{(j)}, rw, pk, sk, r)$ and then computes $(c, K_1) \leftarrow \text{Encap}(pk)$ and $\psi \leftarrow \text{AE.Enc}(H_2(K_0), 0)$. Accordingly, when its partnered client instance $(C^{(i)}, iid)$ receives this specific ψ (passive attack with ψ), it directly uses K_1 corresponding to ψ (for consistence) and computes $\sigma := H_3(K_0, K_1, sid|\psi)$. Recall that in G_3 , $\psi \leftarrow \text{AE.Enc}(H_2(K_0), c)$ rather than $\psi \leftarrow \text{AE.Enc}(H_2(K_0), 0)$.

Finally, simulator Sim records $(S^{(j)}, iid', K_1)$ in Case I and Case II and records $(S^{(j)}, iid', \perp)$ in Case III.

The difference between G_3 and G_4 lies in the generation of ψ in Case III: $\psi \leftarrow \text{AE.Enc}(H_2(K_0), c)$ in G_3 but $\psi \leftarrow \text{AE.Enc}(H_2(K_0), 0)$ in G_4 . In Case III, $H_2(K_0)$ is uniformly distributed and independent of \mathcal{A} 's view, as shown in G_3 . According to the one-time IND-CCA security of AE and hybrid arguments over the (at most) ℓ ciphertexts of AE, we have

$$|\Pr[G_4 \Rightarrow 1] - \Pr[G_3 \Rightarrow 1]| \leq \ell \cdot \text{Adv}_{\text{AE}}^{\text{ot-cca}}(\mathcal{B}_{\text{AE}}).$$

Note that the reduction needs to query the decryption oracle once to simulate the message σ when it receives ψ generated by \mathcal{A} . This is why we need AK have one-time IND-CCA security.

Now in G_4 , Sim only needs rw to distinguish Case II and pk to simulate the generation of ψ in Case II. Besides Sim also uses pk to generate c, K_1 in Case II, III, and uses sk to generate σ in Case II, III. But it still needs pw to generate (rw, pk, sk) .

Game G_5 (Simulation of Client's Message σ and Session Key sKey Without pw). In G_5 , Sim is the same as in G_4 , except for Sim's generation of σ and sKey for the client instance $(C^{(i)}, iid)$ when receiving ψ . We describe Sim's simulations in the three cases in G_5 .

- **Case I occurs to $(C^{(i)}, iid)$: \mathcal{A} successfully guesses pw by issuing “(Testpw, $C^{(i)}, iid, rw = H_0(pw)$)” query to $\mathcal{F}_{\text{pake}}$.** In this case, Sim can extract the true password pw from $\mathcal{F}_{\text{pake}}$ and simulate the generation of σ in the same way as G_4 . More precisely, Sim first decrypts $c \leftarrow \text{AE.Dec}(H_2(K_0), \psi)$.
- If $c = \perp$, then Sim issues (Abort, $C^{(i)}, iid$) query to $\mathcal{F}_{\text{pake}}$. As a result, $\mathcal{F}_{\text{pake}}$ returns (\perp, \perp) to $(C^{(i)}, iid)$ to reject the session. In this case the session is rejected in both G_4 and G_5 .
- Otherwise, Sim generates $(pk, sk) \leftarrow \text{KeyGen}(H_1(pw))$, decrypts $K_1 \leftarrow \text{Decap}(sk, c)$ and computes $\sigma := H_3(K_0, K_1, sid|\psi)$. Finally, Sim issues (CorruptKey, $C^{(i)}, iid, sid|\psi|\sigma, H_4(K_0)$) query to $\mathcal{F}_{\text{pake}}$ and $\mathcal{F}_{\text{pake}}$ returns $(sid|\psi|\sigma, \text{sKey} = H_4(K_0))$ to $(C^{(i)}, iid)$. In this case $\text{sKey} = H_4(K_0)$ and $\sigma = H_3(K_0, K_1, sid|\psi)$ in both G_4 and G_5 .

- **Case II occurs to $(C^{(i)}, iid)$: \mathcal{A} must have stolen server’s password file and then impersonates party $S^{(j)}$.** In this case, Sim first decrypts $c \leftarrow \text{AE.Dec}(H_2(K_0), \psi)$.
- If $c = \perp$, then Sim issues $(\text{Abort}, C^{(i)}, iid)$ query to $\mathcal{F}_{\text{apake}}$. As a result, $\mathcal{F}_{\text{apake}}$ returns (\perp, \perp) to $(C^{(i)}, iid)$ to reject the session. In this case the session is rejected in both G_4 and G_5 .
- If $c \neq \perp$, then Sim retrieves sk from its trapdoor record $(C^{(i)}, S^{(j)}, rw, pk, sk, r)$ to decrypt $K_1 \leftarrow \text{Decap}(sk, c)$ and computes $\sigma := H_3(K_0, K_1, sid|\psi)$. Then Sim issues $(\text{Impersonate}, C^{(i)}, iid)$ query to $\mathcal{F}_{\text{apake}}$, which indicates that Sim impersonates $S^{(j)}$ to attack $(C^{(i)}, iid)$. Finally, Sim issues $(\text{CorruptKey}, C^{(i)}, iid, sid|\psi|\sigma, H_4(K_0))$ query to $\mathcal{F}_{\text{apake}}$ and then $\mathcal{F}_{\text{apake}}$ must return $(sid|\psi|\sigma, sKey = H_4(K_0))$ to $(C^{(i)}, iid)$. In this case the session key is $sKey = H_4(K_0)$ and $\sigma = H_3(K_0, K_1, sid|\psi)$ in both G_4 and G_5 .
- **Case III: neither Case I nor Case II occurs to $(C^{(i)}, iid)$.** In this case, $K_0, H_2(K_0), H_3(K_0, K_1, c)$ and $H_4(K_0)$ are all simulated with uniform ones by Sim. We further consider the following two subcases according to passive or active attacks.
- **Passive Attack with ψ :** In this case, $(C^{(i)}, iid)$ and $(S^{(j)}, iid')$ must have shared the same PAKE session key K_0 and ψ must be generated by Sim for some instance $(S^{(j)}, iid')$. Sim randomly chosen $\sigma \leftarrow \mathcal{S}\{0, 1\}^\lambda$ and issues $(\text{FreshKey}, C^{(i)}, iid, sid|\psi|\sigma)$ query to $\mathcal{F}_{\text{apake}}$. Then $\mathcal{F}_{\text{apake}}$ returns a uniform session key $sKey \leftarrow \mathcal{S}\{0, 1\}^\lambda$ to $(C^{(i)}, iid)$. Later if the server session $(S^{(j)}, iid')$ receives the same σ later, Sim will issue $(\text{CopyKey}, S^{(j)}, iid', sid|\psi|\sigma)$ query to $\mathcal{F}_{\text{apake}}$ directly. In this case, $(C^{(i)}, iid)$ and $(S^{(j)}, iid')$ share a same uniform session key chosen by $\mathcal{F}_{\text{apake}}$.
- **Active Attack with ψ :** In this case ψ is not generated by Sim or instance $(S^{(j)}, iid')$ and $(C^{(i)}, iid)$ do not share a same PAKE session key. Sim issues $(\text{Abort}, C^{(i)}, iid)$ query to $\mathcal{F}_{\text{apake}}$. As a result, $\mathcal{F}_{\text{apake}}$ returns (\perp, \perp) to $(C^{(i)}, iid)$ to reject the session.

In both sub-cases, Sim does not need to retrieve pk to generate c via $(c, K_1) \leftarrow \text{Encap}(pk)$ for the server instance $(S^{(j)}, iid')$, and it does not need to retrieve sk to decrypt c for client instance $(C^{(i)}, iid)$ any more.

If Case I or Case II occurs, G_4 and G_5 are the same, as analysed above. The difference lies in Case III.

Recall in G_4 , if neither Case I nor Case II occurs, then K_0 is uniform and independent of \mathcal{A} ’s view. Consequently, $H_3(K_0, K_1, sid|\psi)$, $H_4(K_0)$, $H_2(K_0)$ are all uniform to \mathcal{A} and independent of \mathcal{A} ’s view. In the case of passive attack where ψ is generated by Sim, the client message $\sigma := H_3(K_0, K_1, sid|\psi)$ and the client session key $sKey := H_4(K_0)$ are uniform to \mathcal{A} . In G_5 , σ and $sKey$ are uniformly chosen. Therefore, σ and $sKey$ have the same distribution in G_4 and G_5 . In the case of active attack with ψ , Sim directly rejects ψ in G_5 . But in G_4 , Sim accepts if ψ is valid and rejects otherwise. G_5 and G_4 are the same unless the event $\mathbf{Bad}_{\text{Auth}}$ happens, where $\mathbf{Bad}_{\text{Auth}}$ is defined as

Bad_{Auth}: Active attack with ψ results in $\text{AE.Dec}(H_2(K_0), \psi) \neq \perp$ in Case III.

With difference lemma, we know that $|\Pr[G_5 \Rightarrow 1] - \Pr[G_4 \Rightarrow 1]| \leq \Pr[\mathbf{Bad}_{\text{Auth}}]$.

Recall that $H_2(K_0)$ is uniformly distributed in Case III. Thanks to the authenticity of AE, \mathcal{A} only has negligible probability of generating a valid ciphertext ψ without the knowledge of $H_2(K_0)$.

Hence, ψ will always be rejected by $(C^{(i)}, iid)$ except with a negligible probability. Given at most ℓ sessions, we know that $\Pr[\mathbf{Bad}_{\text{Auth}}] \leq \ell \cdot \text{Adv}_{\text{AE}}^{\text{ot-auth}}(\mathcal{B}_{\text{AE}})$. Consequently, we have

$$|\Pr[G_5 \Rightarrow 1] - \Pr[G_4 \Rightarrow 1]| \leq \ell \cdot \text{Adv}_{\text{AE}}^{\text{ot-auth}}(\mathcal{B}_{\text{AE}}).$$

Note that \mathcal{A} may additionally see a valid ciphertext ψ generated by Sim. This is why we need a one-time secure authenticity AE scheme.

Now in G_5 , Sim only needs rw to distinguish Case II, needs pk to simulate the generation of ψ for server instances in Case II, and needs sk to decrypt c so as to compute σ for client instances in Case II. But it still needs pw to generate (rw, pk, sk) .

Game G_6 (Simulation for Password File Without pw). In G_6 , Sim behaves the same as G_5 except for the simulation of generating the password file (rw, pk) .

Recall that in G_5 , Sim uses password pw to generate the password file (rw, pk) upon \mathcal{A} 's StorePWFile query, but reveals the file to \mathcal{A} upon \mathcal{A} 's StealPWFile query. Sim also generates the trapdoor record along with the file but only uses the record for Case II in which StealPWFile must have happened. This fact suggests that Sim can delay the generation of password file (rw, pk) and the trapdoor record $(C^{(i)}, S^{(j)}, rw, pk, sk, H_1(pw))$ until StealPWFile query. This is exactly Sim does without pw in G_6 , but with the help of re-programming technique of ROs.

In G_6 , Sim will keep the password file and the trapdoor record empty until StealPWFile query from \mathcal{A} . We consider three phases of the game.

- **Before receiving** $(\text{StealPWFile}, C^{(i)}, S^{(j)})$ **from** \mathcal{A} : In this phase, the simulations by Sim is exactly the same as that in G_5 . Besides, Sim also does the following. For any random oracle query to $H_0(x)$ or $H_1(x)$ from \mathcal{A} , Sim will also issue a query $(\text{OfflineTestPW}, C^{(i)}, S^{(j)}, x)$ to $\mathcal{F}_{\text{apake}}$ and $\mathcal{F}_{\text{apake}}$ will store an offline-guess record.

Recall that only Case I or Case III happens in this phase. But neither pw nor the trapdoor record is needed by Sim for Case I and Case III in G_5 . So the simulation without pw in G_6 is sound in this phase.

- **When receiving** $(\text{StealPWFile}, C^{(i)}, S^{(j)})$ **from** \mathcal{A} : It must happen that \mathcal{A} has issued a StealPWFile query to a dummy server $S^{(j)}$ and then Sim issues a StealPWFile query to $\mathcal{F}_{\text{apake}}$. According to the specification of $\mathcal{F}_{\text{apake}}$ in Fig. 4, $\mathcal{F}_{\text{apake}}$ sends $(\text{StealPWFile}, C^{(i)}, S^{(j)})$ to Sim.

Upon the output $(\text{StealPWFile}, C^{(i)}, S^{(j)})$ from $\mathcal{F}_{\text{apake}}$,

- (1) If Sim ever issued a $(\text{OfflineTestPW}, \mathcal{C}^{(i)}, \mathcal{S}^{(j)}, x)$ query before such that $x = pw$, $\mathcal{F}_{\text{apake}}$ must have additionally output pw and “correct guess” to Sim . In this case, Sim obtains the correct password pw , and then it can invoke $(pk, sk) \leftarrow \text{KeyGen}(H_1(pw))$, set $rw := H_0(pw)$, return (rw, pk) to \mathcal{A} , and set the trapdoor record as $(\mathcal{C}^{(i)}, \mathcal{S}^{(j)}, rw, pk, sk, H_1(pw))$. The simulations of password file and trapdoor record are exactly like \mathbf{G}_5 .
- (2) Otherwise, Sim randomly samples $rw \leftarrow \$ \{0, 1\}^\lambda$ and $r \leftarrow \$ \mathcal{R}$, invokes $(pk, sk) \leftarrow \text{KeyGen}(r)$, returns (rw, pk) to \mathcal{A} , and stores record $(\mathcal{C}^{(i)}, \mathcal{S}^{(j)}, rw, pk, sk, r)$. In this case, \mathcal{A} did not ever query $H_0(pw)$ or $H_1(pw)$, so these hash values are uniform to \mathcal{A} . Though Sim does not know the value of pw , it implicitly set $H_0(pw) := rw$ and $H_1(pw) = r$. Therefore, in this case, the simulations of password file and trapdoor record are exactly the same from \mathcal{A} 's view no matter in \mathbf{G}_6 or \mathbf{G}_5 .
- **After receiving** $(\text{StealPWFile}, \mathcal{C}^{(i)}, \mathcal{S}^{(j)})$ from \mathcal{A} : In this phase, Sim will use the trapdoor record for the simulations, exactly like in \mathbf{G}_5 . Besides, Sim also keeps an eye on random oracle queries: For each new random oracle query for $H_0(x)$ or $H_1(x)$, Sim will issue a query $(\text{OfflineTestPW}, \mathcal{C}^{(i)}, \mathcal{S}^{(j)}, x)$ to $\mathcal{F}_{\text{apake}}$ and check the reply. If $\mathcal{F}_{\text{apake}}$ returns “correct guess”, Sim will retrieve the record $(\mathcal{C}^{(i)}, \mathcal{S}^{(j)}, rw, pk, sk, r)$ and reprogram $H_0(x = pw) := rw$ and $H_1(x = pw) := r$ by storing (x, rw) in list \mathcal{L}_{H_0} and (x, r) in list \mathcal{L}_{H_1} . As long as \mathcal{A} issues oracle queries on pw , Sim will detect it and obtains the correct password pw . In this way, Sim keeps the consistence between pw and the trapdoor record. Even if \mathcal{A} 's offline-attack succeeds, Sim still give a perfect simulation for \mathcal{A} , just like \mathbf{G}_5 .

Due to the simulation strategy and reprogramming technique, the distribution of trapdoor record $(\mathcal{C}^{(i)}, \mathcal{S}^{(j)}, rw, pk, sk, r = H_1(pw))$ in \mathbf{G}_6 and \mathbf{G}_5 are exactly the same. Therefore, we have

$$\Pr[\mathbf{G}_6 \Rightarrow 1] = \Pr[\mathbf{G}_5 \Rightarrow 1].$$

Game \mathbf{G}_7 (Simulation of Dealing with σ for Server Instances Without pw). In \mathbf{G}_7 , Sim is the same as in \mathbf{G}_6 , except for the simulation of the server instance $(\mathcal{S}^{(j)}, iid')$ when receiving σ . We still consider Sim 's simulations in the three cases in \mathbf{G}_7 .

First, Sim retrieves record $(\mathcal{S}^{(j)}, iid', K_1)$ (that was stored when ψ is simulated for $(\mathcal{S}^{(j)}, iid')$) and the corresponding PAKE key K_0 .

Case I occurs to $(\mathcal{S}^{(j)}, iid')$. If $\sigma \neq H_3(K_0, K_1, sid|\psi)$, then Sim sends $(\text{Abort}, \mathcal{S}^{(j)}, iid')$ to $\mathcal{F}_{\text{apake}}$. As a result, $\mathcal{F}_{\text{apake}}$ returns (\perp, \perp) to $(\mathcal{S}^{(j)}, iid')$ to reject the session. If $\sigma = H_3(K_0, K_1, sid|\psi)$, then Sim sends $(\text{CorruptKey}, \mathcal{S}^{(j)}, iid', sid|\psi|\sigma, H_4(K_0))$ to $\mathcal{F}_{\text{apake}}$. Then $\mathcal{F}_{\text{apake}}$ returns $H_4(K_0)$ to $(\mathcal{S}^{(j)}, iid')$.

Recall in \mathbf{G}_6 , $s\text{Key} = \perp$ if σ is invalid and $s\text{Key} = H_4(K_0)$ if σ is valid. Therefore, \mathbf{G}_6 and \mathbf{G}_7 are the same in this case.

Case II occurs to $(\mathcal{S}^{(j)}, iid')$. If $\sigma \neq H_3(K_0, K_1, sid|\psi)$ or $\sigma = H_3(K_0, K_1, sid|\psi)$ but $\mathcal{F}_{\text{apake}}$ did not return “correct guess” to any of Sim 's $(\text{OfflineTestPW},$

$(C^{(i)}, S^{(j)}, pw')$ queries, then Sim sends $(\text{Abort}, S^{(j)}, iid')$ to $\mathcal{F}_{\text{apake}}$. As a result, $\mathcal{F}_{\text{apake}}$ returns (\perp, \perp) to $(S^{(j)}, iid')$ to reject the session.

If $\sigma = H_3(K_0, K_1, sid|\psi)$ and $\mathcal{F}_{\text{apake}}$ returned “correct guess” to one of Sim ’s $(\text{OfflineTestPW}, C^{(i)}, S^{(j)}, pw')$ queries, then Sim sends $(\text{CorruptKey}, S^{(j)}, iid', sid|\psi|\sigma, H_4(K_0))$ to $\mathcal{F}_{\text{apake}}$. Accordingly $\mathcal{F}_{\text{apake}}$ returns $(sid|\psi|\sigma, sKey := H_4(K_0))$ to $(S^{(j)}, iid')$.

Recall in G_6 , as long as Case II occurs, the simulator will set the session key $sKey := H_4(K_0)$ if $\sigma = H_3(K_0, K_1, sid|\psi)$, and set $sKey := \perp$ if $\sigma \neq H_3(K_0, K_1, sid|\psi)$.

Case III occurs to $(S^{(j)}, iid')$. We further consider whether σ is from \mathcal{A} ’ passive attack or active attack.

- **Passive Attack with σ :** In this case, there must exist some instance $(C^{(i)}, iid)$ which has agreed PAKE key K_0 with $(S^{(j)}, iid')$, and σ must be generated by Sim for $(C^{(i)}, iid)$. Moreover, Sim must also have issued $(\text{FreshKey}, C^{(i)}, iid, sid|\psi|\sigma)$ query to $\mathcal{F}_{\text{apake}}$ and $\mathcal{F}_{\text{apake}}$ returns a uniform session key $sKey$ to $(C^{(i)}, iid)$. Now Sim issues $(\text{CopyKey}, S^{(j)}, iid', sid|\psi|\sigma)$ query to $\mathcal{F}_{\text{apake}}$ directly. In this case, $(C^{(i)}, iid)$ and $(S^{(j)}, iid')$ share a same uniform session key $sKey$ chosen by $\mathcal{F}_{\text{apake}}$ and the session key $sKey$ is simulated with the help of $\mathcal{F}_{\text{apake}}$, without the knowledge of pw or the trapdoor record, just like G_6 .
- **Active Attack with ψ :** Sim sends $(\text{Abort}, S^{(j)}, iid')$ to $\mathcal{F}_{\text{apake}}$, and accordingly $\mathcal{F}_{\text{apake}}$ returns $(\perp, sKey := \perp)$ to $(S^{(j)}, iid')$ to reject the session. Recall that in Case III of G_6 , $H_4(K_0, K_1, sid|\psi)$ is uniformly distributed and independent of \mathcal{A} ’s view, the simulator will reject the session by setting $sKey := \perp$ except with negligible probability.

Therefore, G_6 and G_7 are the same in Case III except with negligible probability.

According to the above analyses, G_7 and G_6 are the same except a bad event **Bad** happens, where

Bad: Case II occurs to some $(S^{(j)}, iid')$, $\sigma = H_3(K_0, K_1, sid|\psi)$, but until then none of the Sim ’s $(\text{OfflineTestPW}, C^{(i)}, S^{(j)}, pw')$ queries results in “correct guess”.

With difference lemma, we know that $|\Pr[G_7 \Rightarrow 1] - \Pr[G_6 \Rightarrow 1]| \leq \Pr[\mathbf{Bad}]$. Next we show a reduction algorithm \mathcal{B}_{KEM} and prove $\Pr[\mathbf{Bad}] \leq N^{2\ell} \cdot \text{Adv}_{\text{KEM}}^{\text{ow-pca}}(\mathcal{B}_{\text{KEM}})$.

\mathcal{B}_{KEM} obtains a public key pk^* , a key encapsulation c^* , and has access to oracle $\text{Check}(\cdot, \cdot)$ which on input (c, K) returns 1 iff $\text{Decap}(sk^*, c) = K$. \mathcal{B}_{KEM} aims to find K^* s.t. $\text{Decap}(sk^*, c^*) = K^*$.

In the reduction, \mathcal{B}_{KEM} plays the role of the simulator Sim in G_7 . It first randomly choose $(C^{(*)}, S^{(*)}, iid') \leftarrow \$ [N] \times [N] \times [\ell]$. \mathcal{B}_{KEM} sets $(S^{(*)}, iid')$ as the target instance. Suppose that $(C^{(*)}, iid)$ is the partnered instance. \mathcal{B}_{KEM} will detect whether **Bad** happen on $(S^{(*)}, iid')$.

- For any other instances, \mathcal{B}_{KEM} behaves just like Sim does in G_7 .

– For the instances $(C^{(*)}, iid)$ and $(S^{(*)}, iid')$, \mathcal{B}_{KEM} does the simulations as follows.

- **Before receiving** $(\text{StealPWFile}, C^{(*)}, S^{(*)})$ **from** \mathcal{A} : \mathcal{B}_{KEM} behaves just like Sim does in G_7 .
- **Upon receiving** $(\text{StealPWFile}, C^{(*)}, S^{(*)})$ **from** \mathcal{A} : \mathcal{B}_{KEM} issues a StealPWFile query to $\mathcal{F}_{\text{apake}}$. If $\mathcal{F}_{\text{apake}}$ replies with pw and “correct guess”, then **Bad** does not happen on $(S^{(*)}, iid')$, and \mathcal{B}_{KEM} aborts the game. Otherwise, \mathcal{B}_{KEM} randomly samples $rw \leftarrow \mathcal{S}\{0, 1\}^\lambda$, sets the trapdoor record as $(C^{(*)}, S^{(*)}, rw, pk^*, sk = ?, r = ?)$, and returns the password file (rw, pk^*) to \mathcal{A} .
- **After receiving** $(\text{StealPWFile}, C^{(*)}, S^{(*)})$ **from** \mathcal{A} : During this phase, for any offline attack from \mathcal{A} , if the corresponding $(\text{OfflineTestPW}, C^{(*)}, S^{(*)}, pw')$ query to $\mathcal{F}_{\text{apake}}$ results in “correct guess”, then **Bad** does not happen on $(S^{(*)}, iid')$ and \mathcal{B}_{KEM} aborts the game.

For the session between instances $(C^{(*)}, iid)$ and $(S^{(*)}, iid')$, \mathcal{B}_{KEM} first simulates the generation of PAKE key K_0 without pw and trapdoor record, just like Sim . If Case II does not happen, then **Bad** does not happen on $(S^{(*)}, iid')$ and \mathcal{B}_{KEM} aborts the game.

Next, for server instance $(S^{(*)}, iid')$, \mathcal{B}_{KEM} can simulate the generation of ψ , \mathcal{B}_{KEM} invokes $\psi \leftarrow \text{AE.Enc}(H_2(K_0), c^*)$. Since c^* is an encapsulation under pk^* , we know \mathcal{B}_{KEM} 's simulation is perfect, just like Sim .

For client instance $(C^{(*)}, iid)$ to deal with ψ' , recall that Sim may decrypt ψ' with sk^* to generate K'_1 and then σ , but \mathcal{B}_{KEM} has no sk^* at all. To deal with this problem, \mathcal{B}_{KEM} resorts to ROs and the re-programming techniques: upon $(C^{(*)}, iid)$ receiving ψ' , \mathcal{B}_{KEM} invokes $c \leftarrow \text{AE.Dec}(H_2(K_0), \psi')$. If $c = \perp$, \mathcal{B}_{KEM} behaves just like Sim (no sk^* involved). If $c \neq \perp$, for all hash queries $H_3(K_0, K'_1, sid|\psi')$, \mathcal{B}_{KEM} first checks whether there exists K'_1 such that $\text{Check}(c, K'_1) = 1$. If yes, set $\sigma := H_3(K_0, K'_1, sid|\psi')$. Otherwise randomly choose $\sigma \leftarrow \mathcal{S}\{0, 1\}^\lambda$, and set $H_3(K_0, K'_1 = ?, sid|\psi') := \sigma$ with $?$ denoting the undermined value of K'_1 . It then returns σ to \mathcal{A} and returns $s\text{Key} := H_2(K_0)$ as the session key. If later \mathcal{A} issues any new query $H_3(K_0, K'_1, sid|\psi')$, \mathcal{B}_{KEM} checks whether $\text{Check}(c, K'_1) = 1$. If yes, \mathcal{B}_{KEM} re-programme $H_3(K_0, K'_1, sid|\psi') := \sigma$.

Upon $(S^{(*)}, iid')$ receiving σ' . If there is no H_3 query such that $\sigma' = H_3(K_0, K'_1, sid|\psi)$, then **Bad** does not happen on $(S^{(*)}, iid')$ since σ' is hardly valid due to the uniformity of RO, and then \mathcal{B}_{KEM} aborts the game. Otherwise, find \mathcal{A} 's hash query such that $\sigma' = H_3(K_0, K'_1, sid|\psi)$, \mathcal{B}_{KEM} checks whether $\text{Check}(c, K'_1) = 1$. If no, then $K'_1 \neq K_1^* := \text{Decap}(sk^*, c)$ so $\sigma' \neq H_3(K_0, K_1^*, sid|\psi)$. Thus **Bad** does not happen on $(S^{(*)}, iid')$ and \mathcal{B}_{KEM} aborts the game. If yes, $\text{Check}(c, K'_1) = 1$ implies $K'_1 = K_1^* = \text{Decap}(sk^*, c)$. Now **Bad** happens, \mathcal{B}_{KEM} just replies this K'_1 as it answer to its OW-PCA challenger.

The above description and analysis shows that \mathcal{B}_{KEM} presents a perfect simulation like **Sim**. As long as **Bad** happens on $(S^{(*)}, iid')$, \mathcal{B}_{KEM} wins in its OW-PCA experiment. Consequently, we have

$$\text{Adv}_{\text{KEM}}^{\text{ow-pca}}(\mathcal{B}_{\text{KEM}}) = \Pr[\mathbf{Bad} \text{ on } (S^{(*)}, iid')] = \frac{1}{N^2\ell} \cdot \Pr[\mathbf{Bad}].$$

So,

$$|\Pr[G_7 \Rightarrow 1] - \Pr[G_6 \Rightarrow 1]| \leq \Pr[\mathbf{Bad}] = N^2\ell \cdot \text{Adv}_{\text{KEM}}^{\text{ow-pca}}(\mathcal{B}_{\text{KEM}}).$$

Finally, by combining all the statements across G_0 - G_7 , we get

$$\begin{aligned} |\Pr[\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}] - \Pr[\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}}]| &\leq N^2\ell \cdot \text{Adv}_{\text{KEM}}^{\text{ow-pca}}(\mathcal{B}_{\text{KEM}}) + \frac{q^2 + q + 1}{2^\lambda} \\ &\quad + \ell \cdot \text{Adv}_{\text{AE}}^{\text{ot-auth}}(\mathcal{B}_{\text{AE}}) + \ell \cdot \text{Adv}_{\text{AE}}^{\text{ot-cca}}(\mathcal{B}_{\text{AE}}). \end{aligned}$$

□

Remark 2 (Tightly secure aPAKE compiler.) Note that in the security proof, the security loss is due to the guessing strategy in the reduction between G_7 and G_6 , which relies on the OW-PCA security of the KEM scheme. By imposing stronger security on KEM, it is possible to make the reduction a tight one. For example, Pan et al. [27] proposed the OW-ChCCA security for KEM and presented instantiation of such a KEM with tight OW-ChCCA security based on the (Matrix) DDH assumption. The OW-ChCCA security considers a multi-user setting, where besides the check oracle, the adversary adaptively corrupts users to obtain their secret keys, gets multiple challenge ciphertext encapsulations $\{c_j\}$, adaptively reveals some ciphertexts $\{c'_i\} \subseteq \{c_j\}$ to obtain encapsulation keys, and adaptively obtains decryption results for $\{c'_k\}$ such that $\{c_j\} \cap \{c'_k\} = \emptyset$, and it requires that it is still hard for such an adversary to guess correctly an encapsulation key for any unrevealed ciphertext under public keys of uncorrupted users.

Theorem 2. *Under the same condition as in Theorem 1, if KEM is replaced with an OW-ChCCA secure one and AE is an information-theoretical AE scheme with one-time authenticity and one-time CCA security, then we have*

$$|\Pr[\mathbf{Real}_{\mathcal{Z}, \mathcal{A}}] - \Pr[\mathbf{Ideal}_{\mathcal{Z}, \text{Sim}}]| \leq \text{Adv}_{\text{KEM}}^{(N^2, \ell)\text{-ChCCA}}(\mathcal{B}_{\text{KEM}}) + \frac{2\ell + q^2 + q + 1}{2^\lambda}.$$

The security proof of Theorem 2 is given in the full version [23]. In Appendix B of [23], we recall the formal definition of OW-ChCCA security for KEM. Then we prove that our compiler is tightness-preserving when equipped with such a OW-ChCCA secure KEM. We also review the specific tightly OW-ChCCA secure KEM based on the Matrix DDH assumption, which was proposed in [27].

4 Instantiations of aPAKE from Our Compiler and PAKE

Our aPAKE compiler needs an OW-PCA secure KEM scheme and an AE scheme with one-time authenticity and one-time CCA security. A good candidate for AE is the canonical information-theoretic construction.

Instantiation of One-Time Secure AE. For one-time secure AE, we present a concrete information-theoretically secure AE scheme $\text{AE}_{\text{it}} = (\text{AE.Enc}, \text{AE.Dec})$ from one-time pad and one-time MAC, using the Encrypt-then-MAC approach. Here the key space is $\{0, 1\}^{3\lambda}$, the message space is $\{0, 1\}^\lambda$, and the ciphertext space is $\{0, 1\}^{2\lambda}$. All operations are over field \mathbb{F}_{2^λ} .

- $\text{AE.Enc}(k, m)$: Parse $k := (k_1, k_2, k_3) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}$ and $m \in \mathbb{F}_{2^\lambda}$. Compute $c_1 := k_1 + m$ and $c_2 = k_2 \times c_1 + k_3$.
- $\text{AE.Dec}(k, c)$: Parse $k := (k_1, k_2, k_3) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}$ and $c = (c_1, c_2)$. If $c_2 \neq k_2 \times c_1 + k_3$: return \perp . Otherwise return $c_1 - k_1$.

Lemma 1. *For the above AE, we have*

$$\text{Adv}_{\text{AE}}^{\text{ot-auth}}(\mathcal{A}) \leq 1/2^\lambda, \quad \text{Adv}_{\text{AE}}^{\text{ot-cca}}(\mathcal{A}) \leq 1/2^\lambda.$$

for any (even all-powerful) adversaries.

The security proof of Lemma 1 is shown in the full version [23].

The OW-PCA security is a security notion weaker the CCA security, which admits flexible choices for the underlying KEM in our compiler. Next we show how to instantiate our compiler according to the PAKE scheme so as the resulting aPAKE scheme enjoying good features.

4.1 Most Efficient aPAKE from Lattice

Recall that there exists efficient UC-secure PAKE protocols [2, 20, 30] from lattice, where the PAKE protocol in [2, 20] is based on Kyber [7] and that in [30] is based on Saber [10]. Kyber [7] is the NIST PQC winner for KEM, and its CCA security is based on the module-LWE assumption [21], so we choose the Kyber-based UC-secure PAKE scheme [2].

Now we also take Kyber (whose CCA security naturally implies OW-PCA security) as the KEM in our compiler. Then together with the information-theoretically secure AE_{it} , we obtain a Kyber-based aPAKE scheme (A more detailed description of the scheme is provided in the full version [23]).

Note that the second to last round of our compiler can be merged in the last round of PAKE, resulting in a 3-round Kyber-based aPAKE scheme with UC-security.

Comparison to Other Lattice-Based aPAKEs. Up to now, the only approach to lattice-based UC-secure aPAKE is via the signature-aided compiler proposed in [15]. However, signature schemes from lattice, like Dilithium or Falcon, are far less efficient than their KEM counterpart like Kyber. We compare our Kyber-based compiler to the Dilithium-based Compiler in [15] and the Falcon-based Compiler in [15] in terms of the computing and communication efficiency in Table 1.

The comparison in Table 1 suggests that our Kyber-based compiler is the most efficient one, and hence the resulting Kyber-based aPAKE scheme

(=Kyber-based compiler + Kyber-based PAKE) is the *most efficient* aPAKE scheme with UC-security from lattice up to now.

Corollary 1. *Under the same condition as in Theorem 1, suppose that KEM is instantiated with the CCA secure KEM scheme Kyber [7] and AE is an information-theoretical AE scheme with one-time authenticity and one-time CCA security in our compiler. If the underlying PAKE is instantiated with the UC-secure Kyber-based PAKE protocol in [2], then the resulting aPAKE has UC-security s.t.*

$$|\Pr[\mathbf{Real}_{\mathcal{Z},\mathcal{A}}] - \Pr[\mathbf{Ideal}_{\mathcal{Z},\text{Sim}}]| \leq (5q + 2q\ell + 2N^2\ell) \cdot \text{Adv}_{k+1,k,\chi}^{\text{mlwe}}(\mathcal{A}) + 2^{-\Omega(\lambda)},$$

where $\text{Adv}_{k+1,k,\chi}^{\text{mlwe}}(\mathcal{A})$ is the advantage function for the Module-LWE problem [21].

4.2 Tightly Secure aPAKE Scheme from Matrix DDH

For the underlying PAKE protocol, there exist tightly UC-secure PAKE protocols from the CDH assumption [22,28]. Accordingly, our compiler also has tightly secure OW-ChCCA secure KEM [27] from the (Matrix) DDH assumption as candidate.

Now we take the MDDH-based KEM [27] with tight OW-ChCCA security as the KEM in our compiler. Then together with the information-theoretically secure AE_{it} , our compiler can compile the CDH-based tightly UC-secure PAKE scheme to a MDDH-based aPAKE scheme, which serves as the *first tightly UC-secure* aPAKE scheme up to now (The tightly UC-secure aPAKE scheme is provided in [23]).

Corollary 2. *Under the same condition as in Theorem 2, suppose that KEM is instantiated with the tightly OW-ChCCA secure MDDH-based KEM in [27] and AE is an information-theoretical AE scheme with one-time authenticity and one-time CCA security in our compiler. If the underlying PAKE is instantiated with the tightly UC-secure CDH-based PAKE protocol in [22], then the resulting aPAKE has tight UC-security such that*

$$|\Pr[\mathbf{Real}_{\mathcal{Z},\mathcal{A}}] - \Pr[\mathbf{Ideal}_{\mathcal{Z},\text{Sim}}]| \leq 12 \cdot \text{Adv}_{\text{MDDH}}(\mathcal{A}) + 2 \cdot \text{Adv}_{\text{CDH}}(\mathcal{A}) + 2^{-\Omega(\lambda)},$$

where $\text{Adv}_{\text{MDDH}}(\mathcal{A})$ and $\text{Adv}_{\text{CDH}}(\mathcal{A})$ are the advantage function for the MDDH problem [13] and the CDH problem.

Acknowledgements. We would like to thank the reviewers for their valuable comments. This work was partially supported by Guangdong Major Project of Basic and Applied Basic Research (2019B030302008), National Natural Science Foundation of China under Grant 61925207 and Grant 62372292, and the National Key R&D Program of China under Grant 2022YFB2701500.

References

1. Barbosa, M., Gellert, K., Hesse, J., Jarecki, S.: Bare PAKE: universally composable key exchange from just passwords. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part II. LNCS, vol. 14921, pp. 183–217. Springer (2024), https://doi.org/10.1007/978-3-031-68379-4_6
2. Beguinet, H., Chevalier, C., Pointcheval, D., Ricosset, T., Rossi, M.: GeT a CAKE: Generic transformations from key encapsulation mechanisms to password authenticated key exchanges. In: Tibouchi, M., Wang, X. (eds.) ACNS 23, Part II. LNCS, vol. 13906, pp. 516–538. Springer, Heidelberg (Jun 2023). https://doi.org/10.1007/978-3-031-33491-7_19
3. Bellare, S.M., Merritt, M.: Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In: Denning, D.E., Pyle, R., Ganesan, R., Sandhu, R.S., Ashby, V. (eds.) ACM CCS 93. pp. 244–250. ACM Press (Nov 1993). <https://doi.org/10.1145/168588.168618>
4. Benhamouda, F., Pointcheval, D.: Verifier-based password-authenticated key exchange: New models and constructions. Cryptology ePrint Archive, Report 2013/833 (2013), <https://eprint.iacr.org/2013/833>
5. Bernstein, D.J., Hülsing, A., Kölbl, S., Niederhagen, R., Rijneveld, J., Schwabe, P.: The SPHINCS⁺ signature framework. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2129–2146. ACM Press (Nov 2019). <https://doi.org/10.1145/3319535.3363229>
6. Boneh, D., Dagdelen, Ö., Fischlin, M., Lehmann, A., Schaffner, C., Zhandry, M.: Random oracles in a quantum world. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 41–69. Springer, Heidelberg (Dec 2011). https://doi.org/10.1007/978-3-642-25385-0_3
7. Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-kyber: a cca-secure module-lattice-based kem. In: 2018 IEEE European Symposium on Security and Privacy (EuroS&P). pp. 353–367. IEEE (2018)
8. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001). <https://doi.org/10.1109/SFCS.2001.959888>
9. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 404–421. Springer, Heidelberg (May 2005). https://doi.org/10.1007/11426639_24
10. D’Anvers, J.P., Karmakar, A., Roy, S.S., Vercauteren, F.: Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In: Joux, A., Nitaj, A., Rachidi, T. (eds.) AFRICACRYPT 18. LNCS, vol. 10831, pp. 282–305. Springer, Heidelberg (May 2018). https://doi.org/10.1007/978-3-319-89339-6_16
11. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Dilithium: A lattice-based digital signature scheme. IACR TCHES **2018**(1), 238–268 (2018). <https://doi.org/10.13154/tches.v2018.i1.238-268>, <https://tches.iacr.org/index.php/TCHES/article/view/839>
12. Duman, J., Hartmann, D., Kiltz, E., Kunzweiler, S., Lehmann, J., Riepel, D.: Generic models for group actions. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part I. LNCS, vol. 13940, pp. 406–435. Springer, Heidelberg (May 2023). https://doi.org/10.1007/978-3-031-31368-4_15

13. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1_8
14. Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z., et al.: Falcon: Fast-fourier lattice-based compact signatures over ntru. Submission to the NIST’s post-quantum cryptography standardization process **36**(5), 1–75 (2018)
15. Gentry, C., MacKenzie, P., Ramzan, Z.: A method for making password-based key exchange resilient to server compromise. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 142–159. Springer, Heidelberg (Aug 2006). https://doi.org/10.1007/11818175_9
16. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008). <https://doi.org/10.1145/1374376.1374407>
17. Gu, Y., Jarecki, S., Krawczyk, H.: KHAPE: Asymmetric PAKE from key-hiding key exchange. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 701–730. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84259-8_24
18. Hesse, J.: Separating symmetric and asymmetric password-authenticated key exchange. In: Galdi, C., Kolesnikov, V. (eds.) SCN 20. LNCS, vol. 12238, pp. 579–599. Springer, Heidelberg (Sep 2020). https://doi.org/10.1007/978-3-030-57990-6_29
19. Hwang, J.Y., Jarecki, S., Kwon, T., Lee, J., Shin, J.S., Xu, J.: Round-reduced modular construction of asymmetric password-authenticated key exchange. In: Catalano, D., De Prisco, R. (eds.) SCN 18. LNCS, vol. 11035, pp. 485–504. Springer, Heidelberg (Sep 2018). https://doi.org/10.1007/978-3-319-98113-0_26
20. Januzelli, J., Roy, L., Xu, J.: Under what conditions is encrypted key exchange actually secure? Cryptology ePrint Archive, Paper 2024/324 (2024), <https://eprint.iacr.org/2024/324>
21. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. DCC **75**(3), 565–599 (2015). <https://doi.org/10.1007/s10623-014-9938-4>
22. Liu, X., Liu, S., Han, S., Gu, D.: EKE meets tight security in the Universally Composable framework. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part I. LNCS, vol. 13940, pp. 685–713. Springer, Heidelberg (May 2023). https://doi.org/10.1007/978-3-031-31368-4_24
23. Lyu, Y., Liu, S., Han, S.: Efficient asymmetric PAKE compiler from KEM and AE. Cryptology ePrint Archive, Paper 2024/1400 (2024), <https://eprint.iacr.org/2024/1400>
24. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_43
25. McQuoid, I., Xu, J.: An efficient strong asymmetric pake compiler instantiable from group actions. In: ASIACRYPT 2023. pp. 176–207. Springer (2023), <https://eprint.iacr.org/2023/1434>
26. Okamoto, T., Pointcheval, D.: REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 159–175. Springer, Heidelberg (Apr 2001). https://doi.org/10.1007/3-540-45353-9_13

27. 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
28. Pan, J., Zeng, R.: A generic construction of tightly secure password-based authenticated key exchange. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part VIII. LNCS, vol. 14445, pp. 143–175. Springer, Heidelberg (Dec 2023). https://doi.org/10.1007/978-981-99-8742-9_5
29. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC. pp. 84–93. ACM Press (May 2005). <https://doi.org/10.1145/1060590.1060603>
30. Santos, B.F.D., Gu, Y., Jarecki, S.: Randomized half-ideal cipher on groups with applications to UC (a)PAKE. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part V. LNCS, vol. 14008, pp. 128–156. Springer, Heidelberg (Apr 2023). https://doi.org/10.1007/978-3-031-30589-4_5
31. Santos, B.F.D., Gu, Y., Jarecki, S., Krawczyk, H.: Asymmetric PAKE with low computation and communication. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 127–156. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_5
32. Schmidt, J.: Requirements for password-authenticated key agreement (pake) schemes. Tech. rep. (2017), <https://tools.ietf.org/html/rfc8125>
33. Shoup, V.: Security analysis of itSPAKE2+. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part III. LNCS, vol. 12552, pp. 31–60. Springer, Heidelberg (Nov 2020). https://doi.org/10.1007/978-3-030-64381-2_2



Threshold PAKE with Security Against Compromise of All Servers

Yanqi Gu¹, Stanislaw Jarecki¹, Pawel Kedzior², Phillip Nazarian¹,
and Jiayu Xu³

¹ University of California Irvine, Irvine, USA

{yanqig1,sjarecki,pnazaria}@uci.edu

² University of Warsaw, Warsaw, Poland

p.kedzior@mimuw.edu.pl

³ Oregon State University, Corvallis, USA

xujiay@oregonstate.edu

Abstract. We revisit the notion of *threshold Password-Authenticated Key Exchange* (tPAKE), and we extend it to *augmented* tPAKE (atPAKE), which protects password information even in the case all servers are compromised, except for allowing an (inevitable) offline dictionary attack. Compared to prior notions of tPAKE this is analogous to replacing symmetric PAKE, where the server stores the user’s password, with an *augmented* (or *asymmetric*) PAKE, like OPAQUE [43], where the server stores a password *hash*, which can be used only as a target in an offline dictionary search for the password. An atPAKE scheme also strictly improves on the security of an aPAKE, by secret-sharing the password hash among a set of servers. Indeed, our atPAKE protocol is a natural realization of *threshold OPAQUE*.

We formalize atPAKE in the framework of Universal Composability (UC), and show practical ways to realize it. All our schemes are generic compositions which interface to any aPAKE used as a sub-protocol, making them easier to adopt. Our main scheme relies on *threshold Oblivious Pseudorandom Function* (tOPRF), and our independent contribution fixes a flaw in the UC tOPRF notion of [40] and upgrades the tOPRF scheme therein to achieve the fixed definition while preserving its minimal cost and round complexity. The technique we use enforces implicit agreement on arbitrary context information within threshold computation, and it is of general interest.

1 Introduction

Passwords remain a dominant method of authentication of end-users on the Internet (and beyond),¹ and for decades the prime mechanism for client-server password authentication has been “password-over-TLS”, where the user sends the password to the server over a secure channel authenticated via a Public Key Infrastructure (PKI), and the server compares the received password against the *salted*, i.e. randomized, *password hash* created during user registration.

¹ See e.g. [51, 54, 55] for usability review of alternatives to password authentication.

This scheme has multiple weaknesses, including password visibility on the server during authentication, accidental storage of passwords (for examples see e.g. [1, 2]), and password leakage if the user falls prey to the so-called “phishing” attack. To improve upon this the Internet Engineering Task Force (IETF) conducted a standardization process for a much stronger mechanism, a (strong) *asymmetric (or augmented) Password-Authenticated Key Exchange* (aPAKE), see e.g. [31, 43] and references there-in, where the server stores a randomized password hash for each user but the authentication protocol does not rely on PKI, except possibly for user registration. An aPAKE scheme eliminates all the above weaknesses of password-over-TLS, limiting attacks to the two unavoidable avenues, namely online password tests, and offline password tests in the case of server compromise. The IETF aPAKE competition chose the OPAQUE protocol [12, 43], which in addition to low computation and communication costs, uses an Authenticated Key Exchange (AKE) protocol as a black-box, which makes it easy to integrate with existing secure channel establishment protocols like TLS 1.3 [36, 60].

Strengthening Password Protection by Server Distribution. The online password tests, where the adversary tests a password by using it to authenticate, can be mitigated by setting limits on the number of unsuccessful authentication attempts.² The exposure to offline password tests after server compromise can be reduced as well, using Multi-Party Computation (MPC) [9, 32]: If the aPAKE server is emulated by n parties via an MPC protocol, then the offline testing attack is enabled only if the adversary corrupts more than some threshold $t < n$ of these parties and reconstructs server data, i.e. the password hash.

Employing generic MPC techniques makes the complexity of such a solution linear in the circuit description of the aPAKE server code, which for OPAQUE involves elliptic curve operations, symmetric ciphers, and CRH hashes, with a resulting circuit with tens of thousands of gates. A natural question is whether there are much more practical solutions to an aPAKE where the server-held password hash is secret-shared across a group of servers, and the offline password tests attack avenue is enabled only after corruption of $t+1$ servers.

Threshold *Symmetric* PAKE. MPC-emulation of one party in a *symmetric* PAKE is known as *threshold PAKE* (tPAKE). It was addressed in many works starting from Ford-Kaliski [28] and Jablon [37], whose proposals included informal security arguments. MacKenzie, Shrimpton, and Jakobsson [52] showed the first tPAKE secure in a game-based model, for arbitrary t , assuming PKI. Gennaro and Raimondo [25] showed a password-only tPAKE for $t < n/3$, and Abdalla et al. [3] improved it to $t < n/2$ in the Random Oracle Model (ROM). Jarecki, Kiayias, and Krawczyk [38] constructed game-based tPAKE for any t using an intermediary tool of *password-protected secret-sharing* (PPSS), a.k.a. *password-authenticated secret-sharing* (PASS), [6, 16, 18, 39, 40]. In addition, several works focus on the case of 2 servers, known as *2PAKE* [10, 14, 18, 42, 46–48, 61, 64], including support for proactive security and universal composability [15, 49, 65].

² Two-factor authentication *could* mitigate on-line password tests as well, but two-factor authentication is not commonly used in that way.

However, in these works the servers MPC-emulate a *symmetric* PAKE, and make no security guarantees after corruption of $t+1$ servers, i.e. the adversary who corrupts that threshold might reconstruct a plaintext password, as opposed to its (salted) hash. Current cryptographic literature thus gives the implementers two incomparable choices for protecting password-related information on the server: They can protect it by storing only password hashes, using either the password-over-TLS method or an aPAKE like OPAQUE, or they can protect it using secret-sharing among n servers and using tPAKE, but the latter choice is worse than the former if the attacker corrupts $t+1$ servers.

Contribution #1: UC Augmented Threshold PAKE Model. We define a tPAKE notion which achieves the best of both worlds, i.e. where the compromise of $t+1$ servers leaks only a salted password hash, which enables offline password tests but does not reveal the password in the clear. We call such scheme an *augmented* threshold PAKE (*atPAKE*), and we define it in the Universal Composability (UC) framework [19], by extending the UC (*strong*) aPAKE notion [43] to the multi-server setting.³

We note that the UC framework for expressing security of password authentication protocols, beginning with the Canetti et al. model for UC PAKE [21], is much stronger than extensions of the game-based PAKE security notion of Bellare-Pointcheval-Rogaway (BPR) [8], because in addition to arbitrary interactions between protocol instances it can capture security in the face of arbitrary password correlations, password mistyping, and arbitrary password information leakage. Indeed, UC security was a requirement in the PAKE/aPAKE competition conducted by the IETF, and if distributing the server should strictly upgrade the security properties of UC aPAKE, then the notion that captures the security of such distribution must extend and strengthen the UC aPAKE notion.

We define the UC atPAKE model in a flexible way, distinguishing between two types of servers: A *target server*, who establishes a secure session with a password-authenticated client, and an *auxiliary server*, who holds a secret-share of the password hash but does not establish a secure session. Similar split of roles was considered for both tPAKE and 2PAKE, e.g. in [3,10,25], with the target server sometimes called a *gateway*. However, these solutions assumed a single party playing the target role and required secure channels between each pair of servers. By contrast, our protocols admit an arbitrary number of target and auxiliary servers, with no prior trust assumptions between them. (However, we imply security *only if* the client is uncorrupted during account initialization.) In our model the auxiliary servers do not learn whether authentication succeeds, but if they need this information, e.g. to implement rate-limiting on unsuccessful authentication attempts, it can be supported if each auxiliary server also plays a target server role.

³ Here we use *aPAKE* to refer to *strong* aPAKE functionality of [43], denoted saPAKE therein, and we use *weak aPAKE* to refer to the original aPAKE functionality of [31]. The weak aPAKE model allows for speeding up the offline attack by precomputation performed before server compromise, while the (strong) aPAKE disallows it.

We view our UC atPAKE model as a major part of our contribution. As a sanity check, we verify that any protocol that realizes the UC atPAKE notion, simplified to the case where auxiliary and target servers are equated, is also secure under the game-based tPAKE notion of MacKenzie, Shrimpton, and Jakobsson [52], upgraded using the real-or-random extension of the BPR game-based PAKE model introduced by Abdalla-Fouque-Pointcheval [4]. Recall that a similar check was done by Canetti et al. [21] who verified that their UC PAKE notion implies the BPR game-based PAKE notion of [8].

Contribution #2: Augmented Threshold PAKE Schemes. We show how to realize the atPAKE notion in a modular way, as a generic composition of universally composable sub-protocols, and we prove security of our schemes under the UC atPAKE definition discussed above. Our main proposal realizes UC atPAKE by generically combining *threshold Oblivious Pseudorandom Function* (tOPRF) [40] with an aPAKE scheme. The variants of this approach include replacing the tOPRF component with *threshold Partially Oblivious PRF* (tPOPRF) [27], or with *augmented Password-Protected Secret-Sharing* (aPPSS) [26]. We include an overview of these variants in Sect. 1.2 below.

Contribution #3: Threshold Oblivious PRF. Oblivious PRF (OPRF) [29] is a 2-party protocol between a server who holds key k of PRF F and a client who holds an argument x , s.t. the client outputs $y = F_k(x)$, but no information on x is leaked to the server. OPRF was defined in the UC framework in [38, 39], and it is realized by the “2HashDH” scheme [39] based on the Gap-OneMore-DH (Gap-OMDH) assumption in a prime-order group. For OPRF’s based on other assumptions see e.g. [5, 22, 38, 45]. Threshold OPRF (tOPRF) [40] replaces the OPRF server with a group that secret-shares the PRF key.

We revisit the UC tOPRF notion of [40].⁴ We point out a subtle flaw in their model which makes it ambiguous, and we propose a revised UC tOPRF definition which strengthens the model and fixes the ambiguity. The original tOPRF protocol of [40], based on secret-sharing of the PRF key in the 2HashDH OPRF of [39], does not seem to be provably secure in the fixed model, as we explain in the technical overview below. However, we show that adding an additional form of blinding to this tOPRF scheme – which adds only a very modest cost to the protocol – lets the modified protocol realize the fixed UC tOPRF notion, under the Gap-OMDH and DDH assumptions in ROM. Moreover, whereas the original tOPRF of [40] was analyzed under a complex interactive assumption, which was shown secure in the generic group model, amending that protocol with our blinding method not only lets the protocol realize a stronger security model, but it does so under much simpler security assumptions, adding only the DDH assumption to the Gap-OMDH assumption required by the underlying 2HashDH OPRF.

We extend the UC tOPRF model and our protocol to threshold *Partially Oblivious* PRF (tPOPRF) [27]. This protocol variant has applications e.g. to

⁴ Variants of UC tOPRF notion were also defined and constructed in [7] and [23], but these works target only the setting of n-out-of-n sharing.

an atPAKE scheme where the auxiliary servers share $O(1)$ -sized state across all user accounts. (We discuss this extension in Sect. 1.2 below.)

UC tOPRF can also be used to implement augmented PPSS (aPPSS) [26] via the simple compiler of [40]. Augmented PPSS has further applications, e.g. to *password-protected cryptosystems* [26]. An advantage of aPPSS built from tOPRF via the compiler of [40] is that if tOPRF is *proactively* secure (see Sect. 1.2 below) then so is the resulting aPPSS. By contrast, the aPPSS construction shown in [26] does not seem easy to proactivize.

1.1 Technical Overview

The idea behind our atPAKE protocol is simple and indeed it has appeared in close variants before. Our tOPRF-based construction and its aPPSS-based modification, are close variants of the game-based tPAKE’s of resp. [40] and [38]. They are also natural threshold extensions of resp. the “server learns first” variant of OPAQUE called OPAQUE’ in [34, 43, 56], and of OPAQUE itself [43]. They can also be seen as extensions of the 2PAKE of [42] to general thresholds. Indeed, all these protocols are natural threshold extensions of the *password-hardening* idea of Ford-Kaliski [28] and Jablon [37], which originated all research on threshold PAKE’s. The idea of [28, 37] is a blueprint for 2PAKE: The client on password pw computes a *hardened password* $\text{rw} = F_k(\text{pw})$ via OPRF with (an auxiliary) server #1 who holds key k , and then uses rw to authenticate to (a target) server #2. If the last step is a (weak) aPAKE instance, *this* scheme was shown as a game-based secure augmented 2PAKE [42], where *augmented* refers to the property that corruption of both servers enables offline password tests but does not leak the cleartext password. Indeed, if PRF F_k is appropriately constructed then leaking key k held by server #1 and $\text{rw} = F_k(\text{pw})$ held by server #2, does not leak argument pw except via brute-force attack, where an evaluation of $F_k(\cdot)$ on each argument requires some fixed amount of computation.

First, observe that using (strong) aPAKE [43] instead of PAKE in the last step strengthens security by eliminating advantages due to precomputation in the offline attack in case of all-parties compromise. Second, make it into a “threshold cryptosystem” by replacing OPRF with a (t, n) -threshold tOPRF involving n auxiliary servers. Finally, rather than use the tOPRF-derived value $\text{rw} = F_k(\text{pw})$ directly, let the user derive \mathcal{T} -specific password as $\text{rw}_{\mathcal{T}} = \text{KDF}_{\text{rw}}(\mathcal{T})$. This forms our tOPRF-based atPAKE construction: The client on password pw computes $\text{rw} = F_k(\text{pw})$ via tOPRF with the auxiliary servers who hold a (t, n) -threshold secret-sharing of key k , and then uses \mathcal{T} -specific value derived from rw in an instance of aPAKE with the target server \mathcal{T} . We prove that this scheme is a UC atPAKE if the tOPRF subprotocol is a UC tOPRF.

UC atPAKE Model. The UC atPAKE model we propose is a threshold extension of the UC (strong) aPAKE model [43], customized to a division of roles of auxiliary and target servers, where the former play the role of “guardians”, who must agree for an authentication instance to go through, while the latter are authentication end-points.

The security properties of our UC atPAKE model can be summarized as follows: If a user creates an account with a target server with an identifier sid and a set of n auxiliary servers, then an authentication attempt against the target server requires a unique password guess and participation of $t+1$ of these auxiliary servers who agree on an sid -dependent authentication attempt. Turning to active attacks against the client, if the attacker plays man-in-the-middle on client interaction with all servers then the security is as in PAKE, i.e. one user authentication instance allows the attacker one on-line password test. However, our model strengthens this basic guarantee s.t. if the adversary is passive in client’s interaction with at least some auxiliary servers then the on-line password test is only possible using password guesses which the attacker used himself in on-line interactions with these servers. In other words, as in the case of the online attack against the target server, the attacker can on-line test a password only if it uses it in an sid -tagged interaction with auxiliary servers. This limits online attacks on the client who fails to authenticate the target server but correctly authenticates the auxiliary servers. Finally, offline password tests are enabled only if the adversary corrupts $t+1$ auxiliary servers *and* the target server, and there is no other avenue for learning information on the password.

Requirements on tOPRF and the Flaw in the tOPRF Model [40]. The above requirements impose the following contract on the tOPRF subprotocol: To get one online password test opportunity, either against the target server or the client, the adversary must engage $t+1$ auxiliary server instances under the target account identifier. We strengthen this contract further, so that all participating auxiliary server instances must run on the same sub-session identifier $ssid$. This allows more flexibility in the applications, e.g. $ssid$ can include context information which must be approved by all these servers (and approved by the target server if the latter is in the auxiliary group).

The tOPRF of [40] is perfectly hiding for the client and does not enforce that the servers agree on a sub-session identifier $ssid$ they use to service an interaction with a user. Indeed, it is unclear how a simulator in that protocol can identify which server executions pertain to a user computing consistently on some fixed password. This was observed by [40], who tried to solve this by letting the ideal tOPRF functionality $\mathcal{F}_{\text{tOPRF}}$ pick an arbitrary subset of sessions with distinct $t+1$ servers which are “utilized” for computing one OPRF value. However, this turns out to create an ambiguous and effectively unrealizable model.

Consider a tOPRF instance with $n = 3$ servers S_1, S_2, S_3 and threshold $t = 1$, so one needs 2 servers to compute $F_k(\cdot)$. Assume that the environment allowed each of servers S_1, S_2, S_3 to engage in tOPRF, and the simulator SIM observes that the adversary computes $F_k(\cdot)$ on some argument x_1 , so SIM sends the “evaluate $F_k(\cdot)$ on x_1 ” request to $\mathcal{F}_{\text{tOPRF}}$, which in the model of [40] must decide which pair of server sessions to utilize for this evaluation. Now, whatever choice it makes, with $2/3$ probability it will not match the subset which the real-world adversary used, e.g. if the latter picks its set of two servers at random. For example, assume the adversary computes $F_k(\cdot)$ on x_1 via interaction with S_1 and S_2 , whereas interaction with S_3 was directed at computing $F_k(\cdot)$ on x_2 , and

assume that functionality $\mathcal{F}_{\text{tOPRF}}$ picked the set of utilized servers badly, e.g. it chose $\{S_2, S_3\}$. Unfortunately, this will prevent correct simulation afterwards. Assume the environment cooperates with the adversary, and after this $F_k(x_1)$ evaluation the environment allows S_1 to engage in one more tOPRF instance. The real-world adversary can evaluate $F_k(\cdot)$ on x_2 by using this last interaction with S_1 together with the interaction with S_3 , but when SIM asks $\mathcal{F}_{\text{tOPRF}}$ for the value of $F_k(\cdot)$ on x_2 , functionality $\mathcal{F}_{\text{tOPRF}}$ is stuck: The only two non-utilized server sessions are two sessions by a single server S_1 , whereas an evaluation of $F_k(\cdot)$ on any new argument requires non-utilized sessions with 2 *different* servers.

Fixing the tOPRF Model of [40]. A natural fix is to change $\mathcal{F}_{\text{tOPRF}}$ s.t. simulator SIM must extract the set of servers which the real-world adversary uses to compute $F_k(\cdot)$ on a single argument. However, the tOPRF protocol of [40] does not seem to enable such simulation. As mentioned above, tOPRF of [40] is a threshold version of 2HashDH OPRF of [39]. In the latter F_k is defined as $F_k(x) = H_3(x, H_1(x)^k)$ where H_1, H_3 are RO hash functions, range of H_1 is a group \mathbb{G} of prime order m , and key k is random in \mathbb{Z}_m . This is a PRF under the CDH assumption on group \mathbb{G} in ROM. In the 2HashDH protocol for oblivious evaluation of this PRF, the client on input x picks $r \leftarrow_{\mathcal{S}} \mathbb{Z}_m$ and sends to the server a *blinded* form of its argument, $a = H_1(x)^r$. The server responds with $b = a^k$, which the client de-blinds and outputs $F_k(x)$ as $H_3(x, b^{1/r})$. In protocol 2HashTDH which is a tOPRF version of this scheme [40], k is (Shamir) secret-shared as (k_1, \dots, k_n) , the client sends a to $t+1$ servers, each S_i responds with $b_i = a^{k_i}$, and the client recovers $H_1(x)^k = b^{k/r} = \prod_i (b_i)^{\lambda_i/r}$ where λ_i 's are interpolation coefficients. However, a malicious client can send $a_i = H_1(x)^{r_i}$, for random r_i , to each S_i , and still recover $H_1(x)^k$ as $\prod_i (b_i)^{\lambda_i/r_i}$. Since each a_i is a random group element, the simulator's view is independent of which a_i 's correspond to the same argument x .

Fixing Protocol 2HashDH: Enforcing Agreement Without Interaction.

Still, an honest sender sends the same a to each server, so if the servers preceded the above tOPRF with a round of agreement on a , *ssid*, this would bind $t+1$ server tOPRF sessions to a single x , *ssid*. This extra round of agreement introduces costs and delays, and might not be easy to implement e.g. if one tOPRF node is the user's own personal device, a cell phone or a USB dongle. We show protocol 3HashTDH, which enforces (a, \textit{ssid}) -binding on $t+1$ server sessions without sacrificing the optimal round-complexity of 2HashTDH, using a form of "label-based blinding", applicable to threshold exponentiation. Namely, in addition to sharing (k_1, \dots, k_n) of key k , the servers hold a random zero-sharing (z_1, \dots, z_n) , i.e. shares of a random t -degree polynomial which evaluates to zero, and using another hash function H_2 onto \mathbb{G} , server S_i on input *ssid* and client's message a set its response to $b_i = a^{k_i} \cdot (H_2(\textit{ssid}, a))^{z_i}$. The client computes $H_1(x)^k$ in the same way, i.e. as $\prod_i (b_i)^{\lambda_i/r} = \prod_i (a)^{\lambda_i k_i/r} \cdot \prod_i (H_2(\textit{ssid}, a))^{\lambda_i z_i/r}$: The first factor evaluates to $a^{k/r} = H(x)^k$ as before, while the second factor evaluates to 1 because z_i 's form a zero-sharing, hence $\sum_i \lambda_i z_i = 0$. If H_2 is an RO hash onto \mathbb{G} then under the DDH assumption, the blinding factors $(H_2(\textit{ssid}, a))^{z_i}$ used by any t servers S_i are indistinguishable from random group elements (after

this threshold the blinding factors are correlated because z_i 's lie on a t -degree polynomial). Consequently, unless $t+1$ servers use the same $(a, ssid)$ these blinding factors mask server's responses, making effective evaluation possible only if $t+1$ servers use the same $(a, ssid)$ pair.

1.2 Protocol Variants, Extensions, and Applications

Auxiliary Servers with Constant-Sized State. We consider several further variants of our main tOPRF+(s)aPAKE construction. Firstly, note that in this tOPRF-based construction the auxiliary servers hold separate tOPRF keys for each user. However, the auxiliary servers can keep only a *single secret-shared key* if we replace tOPRF with threshold *Partially Oblivious PRF* (tPOPFRF) [27]. *Partially Oblivious PRF* (POPFRF) [27] extends OPRF to 2-party evaluation of a PRF whose arguments are pairs $(x_{\text{priv}}, x_{\text{pub}})$, where x_{priv} is a private input of the client while x_{pub} is known to both parties, and the protocol hides only x_{priv} from the server. The ‘‘Pythia’’ POPFRF of [27] is secure under One-More Bilinear-DH (OMBDH) under a game-based definition. For POPFRF's based on other assumptions see e.g. [41, 63].⁵ Threshold POPFRF (tPOPFRF) [27, 41] replaces the POPFRF server with a group that secret-shares the PRF key.

In the atPAKE application replacing tOPRF with tPOPFRF means that a single secret-shared key can be re-used across all user accounts: If the servers set the public tPOPFRF input x_{pub} to an account identifier UID, and the user's private input is $x_{\text{priv}} = \text{pw}$, then the user's output is $\text{rw} = F_k(\text{pw}, \text{UID})$, which by the PRF property of F can be interpreted as $\text{rw} = F'_{k[\text{UID}]}(\text{pw})$ where F' is a PRF and $k[\text{UID}]$ is a user-specific PRF key. We show that a threshold version of the Pythia POPFRF [27], amended by the same blinding technique as above, realizes the UC tPOPFRF functionality in ROM under Gap-OMBDH and the DDH assumption on the target group.

Structured Authentication Data. We also consider a variant where tOPRF is replaced with aPPSS [26], i.e. a protocol that allows the client holding password pw to decrypt and authenticate an arbitrary secret rw which was secret-shared among the auxiliary servers in initialization. (Furthermore, the aPPSS is *augmented* in the same sense as atPAKE, i.e. corruption of $t+1$ servers allows only for an offline dictionary attack against the password.) A benefit of using aPPSS over tOPRF is that aPPSS can be built from (non-threshold) OPRF [26, 39], which might require weaker assumptions, e.g. only GapOMDH [39], or only DDH (with more protocols rounds) [16], or LWE [5]. Moreover, aPPSS based on OPRF was shown to be *adaptively secure* for arbitrary t, n parameters [26] (see below).

⁵ POPFRF can be implemented as $F'_k(x_{\text{priv}}, x_{\text{pub}}) = F_{F'_k(x_{\text{pub}})}(x_{\text{priv}})$ using any OPRF F and PRF F^* , but this generic construction might not have properties like threshold implementation, updatability, or verifiability without x_{pub} -dependent keys [27, 63].

A disadvantage of this protocol variant is that it enables offline password testing attack after compromise of $t+1$ auxiliary servers, without the compromise of a target server, because the aPPSS datastructure already allows for verification of the password guess. Indeed, our aPPSS-based atPAKE scheme can be seen as a threshold counterpart to OPAQUE, where the client authenticates the server-supplied data before using it to authenticate to the server, while our tOPRF-based scheme can be seen as a threshold counterpart to OPAQUE', where the (target) server is the first party that can verify an authentication result. Furthermore, we don't know how to make the OPRF-based aPPSS construction *proactive* (see below).

Adaptive and Proactive Security. We show our t(P)OPRF protocols secure for arbitrary t, n parameters in the *static corruptions* model, i.e. if the environment corrupts all parties at the outset of the protocol. A variant of the same argument shows that these protocols remain secure against *adaptive corruptions*, but only if $\binom{n}{t}$ is polynomial in the security parameter. The same security statements carry to the atPAKE construction instantiated if t(P)OPRF is instantiated as above.

Another benefit of our t(P)OPRF-based atPAKE protocol is that it can be *proactivized*, i.e. made secure against proactive adversary, using standard techniques of distributed secret-sharing randomization [35]. We note that the same techniques do not extend to the aPPSS of [26], which leaves an open question of constructing a practical atPAKE which is both proactive and adaptively secure for arbitrary t, n parameters.

Practical Advantages and Applications. Our UC atPAKE protocol is highly practical: It involves a single round of low bandwidth interaction between the client and $t+1$ auxiliary servers, followed by an aPAKE instance between the client and the target server. The servers do not need to communicate directly, which makes the scheme flexible: The auxiliary servers can be implemented by different commercial entities offering a “password hardening” service, or they can be *user's own devices*, like a USB stick or a cell phone. Our scheme requires *no trust assumptions* (and no secure channels) between auxiliary servers or between the auxiliary servers and the target servers.

We note that for simplicity we present our protocols in non-robust versions, but *robustness* and *verifiability* can be added using well-known inexpensive ROM-based non-interactive zero-knowledge proofs. We note also that our atPAKE uses aPAKE as a black-box, and can interface with any existing target-server aPAKE implementation, like OPAQUE. (Indeed, its variants can interface with TLS-OPAQUE, password-over-TLS, and others, see Sect. 6.)

Our UC atPAKE model assumes that the user knows the list of target servers at initialization, but this is done purely to reduce model complexity, because all our protocol variants allow the user to add more target servers by reconstructing rw and computing new $rw_{\mathcal{T}} = \text{KDF}_{rw}(\mathcal{T})$ values. Indeed, this feature make our atPAKE scheme applicable to a (threshold) *password manager* application, implemented by the auxiliary servers.

Other Related Works. As mentioned above, our atPAKE can be thought of as a *threshold password manager* scheme, where the user recovers service-specific passwords from the master password. Our atPAKE protocols map to this application if the master password is pw , the \mathcal{T} -specific password is $\text{rw}_{\mathcal{T}} = \text{KDF}_{\text{rw}}(\mathcal{T})$, and rw is recovered from pw via either tOPRF, tPOPRF, or PPSS. We note that similar usage of *non-threshold* OPRF or POPRF for outsourced password managers was previously considered e.g. in [27, 58, 59].

In other related work, a *password-protected storage* scheme of [23], a variant of PPSS which allows adaptive addition of records, analyzed a similar solution in the n -out-of- n case, i.e. distributed but not (general) threshold.

Another scheme that uses secret-sharing to protect server-stored password hashes is *Distributed Password Verification* [17, 27, 50, 57]. In these schemes compromise of *permanent storage* of all servers leaks only a salted password hash, as in atPAKE. However, these schemes implement only the verification step in the password-over-TLS authentication, i.e. the secret-shared password hash can be used for secure comparison with a cleartext password candidate, but not for authenticated key exchange of a remote entity holding that password.

Roadmap. Section 2 includes notation and security assumptions used across this work, and a brief overview of universal composability. In Sect. 3 we define UC Threshold Oblivious PRF (tOPRF) and show protocol 3HashTDH which realizes that notion. Section 4 introduces the UC Augmented Threshold PAKE (atPAKE) functionality. Section 5 includes our main construction of secure UC atPAKE from UC tOPRF. Section 6 overviews several variants and extensions of the above construction. Due to space constraint we defer some material to the full version version of this paper [33].

Specifically, in the full version we include a proof that UC atPAKE notion of Sect. 3 implies a game-based T-PAKE, a proof of that tOPRF scheme of Sect. 3 realizes UC tOPRF functionality, a proof that atPAKE construction of Sect. 5 realizes UC atPAKE functionality, an extension of our 3HashTDH tOPRF to the threshold *Partially* Oblivious PRF (tPOPRF) (here we overview this extension in Sect. 3.4), and we present three further variants of the atPAKE protocol: (1) replacing tOPRF with tPOPRF [27], (2) replacing tOPRF with *augmented* password-protected secret sharing (aPPSS) [26], and (3) replacing (strong) aPAKE with a weak aPAKE in the last protocol flow.

2 Preliminaries

Notation. We use τ to denote the security parameter. Given a finite set S , we write $x \leftarrow_{\S} S$ to indicate that x is sampled uniformly at random from S . Throughout the paper we assume function $\text{KDF} : \{0, 1\}^{\tau} \times \{0, 1\}^* \rightarrow \{0, 1\}^{\tau}$ which is a PRF.

We recall the computational assumptions that we use in this paper:

Definition 1. Let (\mathbb{G}, \cdot) be a cyclic group of prime order m with generator g . The Decisional Diffie-Hellman problem (DDH) on \mathbb{G} is to distinguish the following two distributions: $\{(g, g^a, g^b, g^c) : a, b, c \leftarrow_{\S} \mathbb{Z}_m\}$ and $\{(g, g^a, g^b, g^{ab}) : a, b \leftarrow_{\S} \mathbb{Z}_m\}$. The DDH assumption is that any PPT adversary \mathcal{A} can only solve this problem with negligible advantage $\text{negl}(\tau)$.

Definition 2. Let (\mathbb{G}, \cdot) be a cyclic group of prime order m with generator g . The Gap One-More Diffie Hellman problem (GapOMDH) on \mathbb{G} is that, given a vector (y^*, h_1, \dots, h_q) where $h_j \leftarrow_{\S} \mathbb{G}$, and $y^* = g^s$ where $s \leftarrow_{\S} \mathbb{Z}_m$, along with access to oracle $\text{OMDH}(a)$ which returns a^s and oracle $\text{DDH}(y, h, u)$ which returns 1 if and only if (g, y, h, u) is a Diffie-Hellman tuple, \mathcal{A} wins if it outputs a set W of pairs (j, h_j^s) where $|W|$ is strictly greater than the number of (unique) queries \mathcal{A} made to the OMDH oracle. The GapOMDH assumption is that any PPT adversary \mathcal{A} can win this game with only negligible probability $\text{negl}(\tau)$.

Secure Channel. We assume secure channels, i.e. secure and authenticated communication, modeled as a UC functionality $\mathcal{F}_{\text{channel}}$ in Fig. 1. We stress that all our protocols use secure channels *only in initialization*.

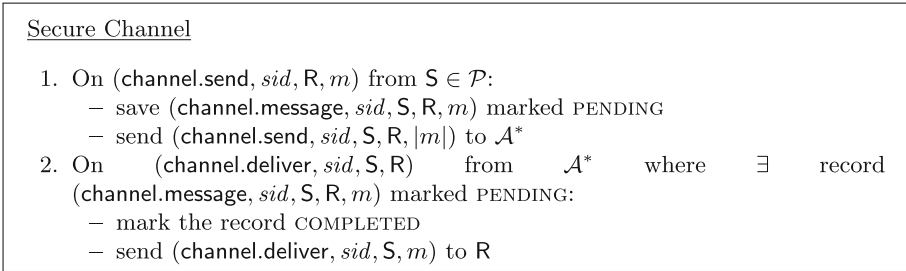


Fig. 1. $\mathcal{F}_{\text{channel}}$: secure and authenticated communication functionality

Universal Composability and Related Notation. In this paper we use the *Universal Composability* (UC) framework [19] to construct security proofs. UC follows the simulation-based paradigm where the security of a protocol is modeled by a machine called the ideal functionality \mathcal{F} , which interacts with a set of “dummy” parties and an ideal world adversary SIM , and does all computation in the ideal world. We say that protocol π securely realizes \mathcal{F} if for any PPT \mathcal{A} , there is a simulator SIM s.t. for all environments \mathcal{Z} , the difference between the real-world view, i.e. an interaction of \mathcal{Z} and \mathcal{A} with parties executing π , and the ideal-world view, i.e. an interaction of \mathcal{Z} and \mathcal{A} with SIM and \mathcal{F} , is negligible in τ .

In the descriptions of ideal functionalities, e.g. Fig. 2, Fig. 4, and others, we specify that each functionality interacts with a set of honest parties \mathcal{P} and an adversary \mathcal{A}^* , and we use notation $\mathcal{P}^* = \mathcal{P} \cup \{\mathcal{A}^*\}$. Each functionality assumes set \mathbf{Corr} includes all initially corrupted parties, and by convention $\mathcal{A}^* \in \mathbf{Corr}$. In our functionalities, protocols, and simulators, we assume strings sid (or $sid_{\mathcal{A}}$, $sid_{\mathcal{T}}$) have the form $sid = (\dots, \mathbf{S})$ where $\mathbf{S} = (S_1, \dots, S_n)$ is a sequence in \mathcal{P} , and \mathbf{S}_{sid} denotes the list \mathbf{S} specified by string sid .

3 Threshold Oblivious PRF

An oblivious pseudorandom function (OPRF) is a protocol with two parties, a server and an evaluator. The server holds the key to a pseudorandom function, and the evaluator holds an input to be evaluated by that pseudorandom function. The OPRF protocol allows the evaluator to evaluate the function obliviously (i.e. without revealing the input to the server) without learning the server’s secret key. Since the PRF key is kept secret, evaluators can only compute the function with the online participation of the server, who could, for example, enforce a rate-limiting policy on evaluators.

3.1 Threshold Oblivious PRF Model

Threshold OPRF (tOPRF) is an extension of OPRF introduced by [40] which distributes server across n parties, s.t. any $t + 1$ of them must participate for an evaluation to succeed. As explained in detail in Sect. 1.1, the tOPRF ideal functionality of [40] has a subtle flaw that seems to make it unrealizable. In Fig. 2 we show $\mathcal{F}_{\text{tOPRF}}$, our modification of the UC tOPRF functionality for the (t, n) threshold. Our functionality $\mathcal{F}_{\text{tOPRF}}$ is a modification of the tOPRF functionality in [40] (see Fig. 1 therein), and it involves several refinements, including the critical fix to the flaw mentioned above. Below we explain the workings of our functionality, including the ways in which it differs from [40].

The Initialization Phase. The initialization is done between an initializer P_0 and a group of n servers \mathbf{S}_{sid} . At the end of the process, each server $S \in \mathbf{S}_{sid}$ is supposed to record a key share, to be used later in evaluation. In the functionality, this is modeled as having a record `toprf.sinit` for each server S , which is marked either `ACTIVE` or `COMPR`; the latter denotes the adversary knowing S ’s key share, which happens if S is compromised, or P_0 is the adversary (in which case the adversary can compute all key shares on its own).

Additionally, P_0 can specify a vector of PRF inputs (x_1, \dots, x_k) , and obtain their PRF outputs $(F_{sid}(x_1), \dots, F_{sid}(x_k))$ during initialization. Though somewhat atypical, this “eval-during-init” feature is natural in our initialization setting. Since P_0 is responsible for creating all key shares and sending them to the servers, there is no reason why P_0 shouldn’t be able to locally evaluate the PRF during initialization. Looking ahead, our tOPRF-based atPAKE construction uses this feature to simplify its initialization phase.

Notation

Initially $\text{tx}[sid, ssid_S, i] := 0$ and $F_{sid}(x)$ is undefined for all $sid, ssid_S, i, x$. When $F_{sid}(x)$ is first referenced $\mathcal{F}_{\text{OPRF}}$ assigns $F_{sid}(x) \leftarrow_{\S} \{0, 1\}^l$.

Initialization

1. On $(\text{toprf.init}, sid, x_1, \dots, x_k)$ from $P_0 \in \mathcal{P}^*$, if sid is new (abort otherwise):
 - send $(\text{toprf.init}, sid, P_0)$ to \mathcal{A}^*
 - send $(\text{toprf.initeval}, sid, F_{sid}(x_1), \dots, F_{sid}(x_k))$ to P_0
 - save $(\text{toprf.init}, sid, P_0)$ and mark it TAMPERED if $P_0 \in \mathbf{Corr}$
2. On $(\text{toprf.sinit}, sid, i, P_0)$ from S where $S = \mathbf{S}_{sid}[i]$ or $(S = \mathcal{A}^*$ and $\mathbf{S}_{sid}[i] \in \mathbf{Corr})$, save record $(\text{toprf.sinit}, sid, i, P_0)$ marked INACTIVE
3. On $(\text{toprf.finit}, sid, i)$ from \mathcal{A}^* where \exists record $\text{urec} = (\text{toprf.init}, sid, P_0)$ and record $\text{srec} = (\text{toprf.sinit}, sid, i, P_0)$ marked INACTIVE:
 - send $(\text{toprf.sinit}, sid, i)$ to $\mathbf{S}_{sid}[i]$
 - if urec is TAMPERED then mark srec TAMPERED
 - else if $\mathbf{S}_{sid}[i] \in \mathbf{Corr}$ then mark srec COMPR
 - else (i.e. urec is not TAMPERED and $\mathbf{S}_{sid}[i] \notin \mathbf{Corr}$) mark srec ACTIVE

Corruption (in the static corruption model disallowed after any other queries)

4. On $(\text{toprf.corrupt}, P)$ from \mathcal{A}^* (with permission from \mathcal{Z}):
 - set $\mathbf{Corr} := \mathbf{Corr} \cup \{P\}$
 - mark every ACTIVE record $(\text{toprf.sinit}, sid, i, P_0)$ COMPR where $P = \mathbf{S}_{sid}[i]$

Evaluation

5. On $(\text{toprf.eval}, sid, ssid_U, x)$ from $U \in \mathcal{P}^*$, if this is the first call from U for sid and $ssid_U$:
 - send $(\text{toprf.eval}, sid, ssid_U, U)$ to \mathcal{A}^*
 - save $(\text{toprf.eval}, sid, ssid_U, U, x)$ marked FRESH
6. On $(\text{toprf.sndrcomplete}, sid, i, ssid_S)$ from S where \exists record $\text{srec} = (\text{toprf.sinit}, sid, i, P_0)$ not marked INACTIVE and $(S = \mathbf{S}_{sid}[i]$ or $(S = \mathcal{A}^*$ and srec is marked COMPR or TAMPERED)):
 - send $(\text{toprf.sndrcomplete}, sid, i, ssid_S)$ to \mathcal{A}^*
 - set $\text{tx}[sid, ssid_S, i]++$
- 6*. On $(\text{toprf.sndrcomplete}^*, sid, i, ssid_S)$ from \mathcal{A}^* where not \exists record $(\text{toprf.init}, sid, P_0)$, set $\text{tx}[sid, ssid_S, i]++$
7. On $(\text{toprf.rcvcomplete}, sid, ssid_U, sid^*, ssid_S^*, \mathbf{C})$ from \mathcal{A}^* where $|\mathbf{C}| = t+1$ and \exists record $(\text{toprf.eval}, sid, ssid_U, U, x)$ marked FRESH:
 - if $\exists j \in \mathbf{C}$ such that $\text{tx}[sid^*, ssid_S^*, j] = 0$, then abort
 - otherwise mark the record COMPLETED, set $\text{tx}[sid^*, ssid_S^*, j]--$ for all $j \in \mathbf{C}$, and send $(\text{toprf.eval}, sid, ssid_U, F_{sid^*}(x))$ to U

Fig. 2. $\mathcal{F}_{\text{OPRF}}$: threshold OPRF functionality, parameterized by threshold t , number of servers n , and output length l .

The Evaluation Phase. In this phase, a user U begins its evaluation by specifying a PRF input x . Any server whose record is ACTIVE, as well as any corrupted

server (whose record is `COMPR`), may choose to participate; each server S is associated with a *ticket counter* $\text{tx}[sid, ssid_S, i]$ (where S is the i -th server in \mathbf{S}_{sid}), which increments if S participates (the mechanics of $ssid_S$ will be explained below). Finally, the ideal adversary specifies an index sid^* which may or may not be the intended index for evaluation sid , together with a set of $t + 1$ servers \mathbf{C} which may or may not be a subset of the intended set of n servers for evaluation \mathbf{S}_{sid} . (Giving the adversary the ability to directly specify the evaluation set \mathbf{C} fixes the flaw in the tOPRF functionality of [40].) After that, \mathbf{U} receives $F_{sid^*}(x)$ provided that none of the servers in \mathbf{C} has ticket counter 0 (w.r.t. index sid^*), and all those ticket counters decrement. This models a man-in-the-middle adversary that might impersonate $t + 1$ servers and let the user evaluate on a “wrong” PRF key.

As a specific case, the ideal adversary can make the user evaluate an unintended function with index $sid^* \neq sid$: the adversary can act as P_0 and init tOPRF sid^* with itself as all n servers; then it can print tickets for servers corresponding to sid^* at will, evaluate function F_{sid^*} locally on its own inputs, and use index sid^* to respond dishonestly to honest evaluators who are intending to evaluate a different function. This all simply represents the real adversary’s ability to locally sample PRFs. To enhance readability, we provide the adversary with a `sndrcomplete*` interface that is simply a shortcut for the above series of actions. This shortcut is never actually needed by the adversary, and indeed the simulator for our tOPRF realization does not make use of it.

The “ticketing mechanism”—inherited from various prior works on (t)OPRF [38, 40]—ensures that in order to compute the PRF value, at least $t + 1$ servers must participate in the evaluation process. In our functionality the action of printing tickets comes from the environment, which models the fact that servers can choose when they wish to participate in an evaluation. In contrast, [40, Fig. 1] models ticket-printing as an adversarial action, effectively reducing tOPRF servers to powerless entities that blindly allow tOPRF evaluations whenever they are asked to.

Our model strengthens the ticketing mechanism by further associating each ticket with the server subsession identifier $ssid_S$ used to print it. To complete an evaluation, the adversary specifies not only the tOPRF instance sid^* and server set \mathbf{C} to use, but also the particular $ssid_S^*$ whose tickets to use up. A successful evaluation requires not only that $t + 1$ servers agree to participate, but that they agree to participate using the same $ssid_S$. These $ssid_S$ values might, therefore, be used to bind tickets to a particular context (for example, an evaluation timestamp). If this feature is unneeded, the $ssid_S$ field can simply be left blank by servers, in which case it will play no role in ticketing.

Server Corruption. Finally, the ideal adversary may corrupt a server P and steal its key share. This is modeled by marking any `ACTIVE topvf.sinit` record for P `COMPR`. Note that the only consequence of corrupting a server is that the adversary can now print tickets for it at will.

3.2 3HashTDH

Figure 3 shows our 3HashTDH protocol $\Pi_{3\text{HashTDH}}$. Protocol $\Pi_{3\text{HashTDH}}$ relies on secure authenticated channels *during initialization*, to allow for secure communication between the initializer party and the n servers participating in the scheme. In Fig. 3 this is modeled via a secure channel functionality $\mathcal{F}_{\text{channel}}$ shown in Fig. 1.

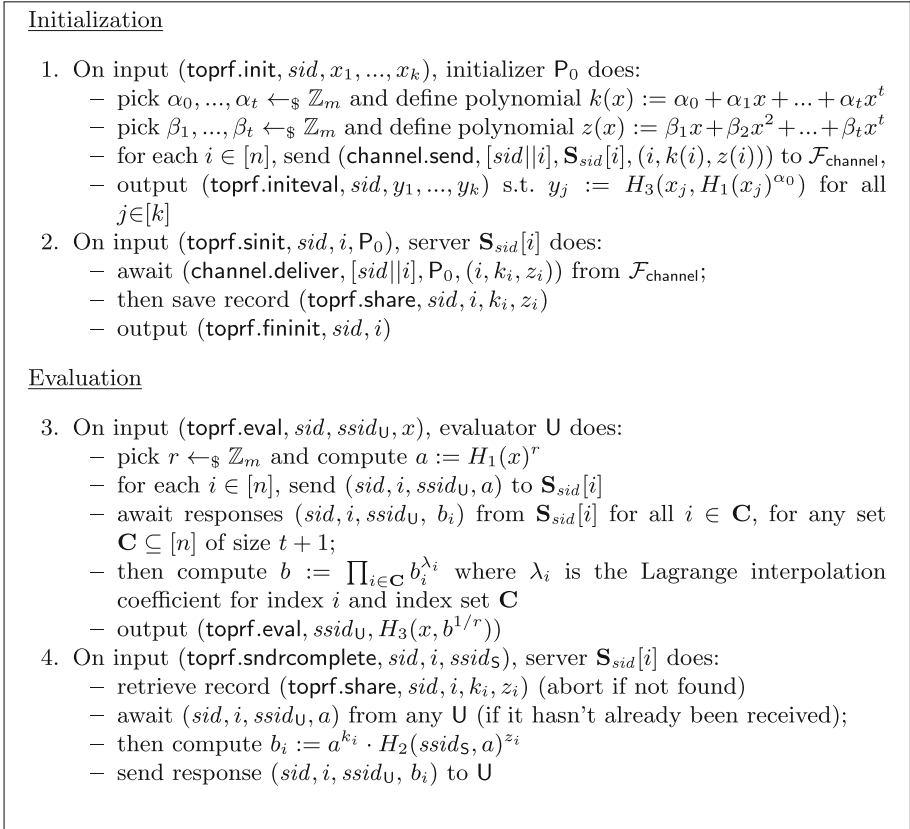


Fig. 3. Protocol $\Pi_{3\text{HashTDH}}$ which realizes $\mathcal{F}_{\text{OPRF}}$ in the $\mathcal{F}_{\text{channel}}$ -hybrid world.

3HashTDH uses a prime-order group \mathbb{G} of size m and three hash functions, $H_1 : \{0, 1\}^* \rightarrow \mathbb{G}$, $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}$, and $H_3 : \{0, 1\}^* \rightarrow \{0, 1\}^l$. The PRF is defined as $F_k(x) = H_3(x, H_1(x)^k)$, exactly the same as the 2HashDH OPRF of [38]. 2HashDH's (single-server) oblivious evaluation protocol for this PRF is the foundation of our protocol. In 2HashDH, the evaluator first picks $r \leftarrow_{\S} \mathbb{Z}_m$ and sends the blinded input $a := H_1(x)^r$ to the server. The server exponentiates using its secret key and sends $b := a^k$ back to the evaluator. The evaluator de-blinds and computes the final hash, outputting $H_3(x, b^{1/r})$.

The 2HashTDH protocol of [40] extends 2HashDH to the threshold, multi-server setting. Each server S_i now holds a (Shamir) secret share k_i of the PRF key k . The evaluator sends $a := H_1(x)^r$ to $t + 1$ servers, and they each respond with $b_i := a^{k_i}$. To combine these responses, the evaluator uses polynomial interpolation in the exponent to compute $b := \prod_i b_i^{\lambda_i} = a^k$, where the λ_i 's are Lagrange interpolation coefficients. Finally the evaluator can compute the final output $H_3(x, b^{1/r})$ as before.

As explained in Sect. 1.1, though, this 2HashTDH protocol does not seem to realize the fixed $\mathcal{F}_{\text{TOPRF}}$ functionality (nor does it realize the non-fixed functionality, which it seems no protocol can realize). To fully simulate the environment's real-world view, the ideal-world simulator must be able to observe exactly which servers the man-in-the-middle adversary is using for each evaluation. If the evaluator honestly sends the same $a = H_1(x)^r$ to all servers, then the simulator can indeed make this observation. However, in 2HashTDH, there is nothing preventing a dishonest evaluator from picking $t + 1$ different blinding exponents r_i and sending information-theoretically random messages $a_i := H_1(x)^{r_i}$ to each server as part of a single evaluation on x .

With 3HashTDH, we aim to fill this simulatability gap by forcing dishonest evaluators to use a single a with all servers for each evaluation, just as honest evaluators do. 3HashTDH accomplishes this by having the servers apply an a -specific blinding factor to their responses; the evaluator can only remove this blinding factor by combining $t + 1$ server responses computed on the same a . In particular, the servers in 3HashTDH hold a Shamir secret sharing (z_1, \dots, z_n) of zero (in addition to their sharing of the PRF key k). Given an evaluator query a , server i responds with $b_i := a^{k_i} \cdot (H_2(a))^{z_i}$. The evaluator combines the responses as $b := \prod_i b_i^{\lambda_i} = \prod_i a^{k_i \lambda_i} \cdot \prod_i (H_2(a))^{z_i \lambda_i}$, the second part of which interpolates to $(H_2(a))^0 = 1$ and disappears. Meanwhile, an adversary who sees only t or less responses for the same a cannot distinguish them from random group elements (under the DDH assumption). Thus, even dishonest evaluators are forced to send the same a to all servers, and simulation succeeds.

As a further feature, we have each server include its subsession identifier $ssid_S$ alongside a in the H_2 input. Then, only server responses corresponding to the same $(ssid_S, a)$ can be combined. Servers can use this $ssid_S$ field to bind evaluations to some context-specific data, which $t + 1$ servers must agree upon in order to have a successful evaluation.

In a concurrent work, Das and Ren [24] have used blinding factors formed in the same way as ours, though for a different purpose: achieving adaptive security for a threshold BLS signature scheme. In their case, the message being signed acts as the “binding data” used as input to the random oracle. Both of our works are preceded by Canetti and Goldwasser [20], who employed a similar blinding factor in a CCA-secure threshold encryption scheme. They also used a Shamir sharing of zero in the exponent, but with a fixed base rather than a random oracle output. Therefore, their scheme requires a fresh zero sharing for each execution.

3.3 Security Analysis of 3HashTDH

If corruptions are static, then the 3HashTDH protocol in Fig. 3 is secure in the random oracle model under the Gap One-More Diffie Hellman (GapOMDH) and Decisional Diffie Hellman (DDH) assumptions.⁶

Theorem 1. *Protocol 3HashTDH realizes functionality $\mathcal{F}_{\text{tOPRF}}$ with parameters t and n in the $\mathcal{F}_{\text{channel}}$ -hybrid model, assuming static corruptions, hash functions H_1 , H_2 , and H_3 modeled as random oracles, and the GapOMDH and DDH assumptions on group \mathbb{G} .*

Specifically, for any efficient adversary \mathcal{A} against protocol 3HashTDH, there exists a simulator SIM such that no efficient environment \mathcal{Z} can distinguish the view of \mathcal{A} interacting with the real 3HashTDH protocol and the view of SIM interacting with the ideal functionality $\mathcal{F}_{\text{tOPRF}}$ with advantage better than $q_I^2/m + q_I \cdot (t \cdot \text{Adv}_{\mathbb{Q}}^{\text{DDH}} + \text{Adv}_{\mathcal{R}}^{\text{GapOMDH}}) \cdot (t+1)$ where q_I is the number of tOPRF instances, $m = |\mathbb{G}|$, and $\text{Adv}_{\mathcal{R}}^{\text{GapOMDH}}$ and $\text{Adv}_{\mathbb{Q}}^{\text{DDH}}$ are bounds on the probability that any efficient algorithm violates the GapOMDH and DDH assumptions, respectively. If corruptions are adaptive, then the 3HashTDH protocol remains secure under the additional assumption that $\binom{n}{t'}$ is a polynomial function of the security parameter for all $0 \leq t' \leq t$.

Theorem 2. *In the case of adaptive corruptions, the statement from Theorem 1 still holds under the additional assumption that $\binom{n}{t'}$ is a polynomial function of the security parameter for all $0 \leq t' \leq t$.*

Specifically, no efficient adversary \mathcal{A} against 3HashTDH has distinguishing advantage better than $q_I^2/m + q_I \cdot (t \cdot \text{Adv}_{\mathbb{Q}}^{\text{DDH}} + \text{Adv}_{\mathcal{R}}^{\text{GapOMDH}}) \cdot \sum_{t'=0}^t \binom{n}{t'}$.

Proof of Theorems 1 and 2 is presented in the full version of the paper. We provide a high-level sketch here.

The essential steps of the proof involve the blinding factors $H_2(ssid_S, a)^{z_i}$ that are applied to the server responses. We construct a GapOMDH reduction that simulates the real 3HashTDH behavior, but picks random key shares to send to the adversary for the corrupted servers (denote the number of corrupted servers as t'). For the uncorrupted servers, the reduction responds to evaluation requests with uniformly random group elements *until* the same $(ssid_S, a)$ is queried to $t - t' + 1$ servers. At that point, the reduction queries the OMDH oracle to find a^k , where k is the OMDH secret (here it acts as the PRF key). The reduction then knows enough values to perform interpolation in the exponent between a^k , the t' key shares attributed to the corrupted servers, and the randomly chosen first $t - t'$ responses. Thus, the $(t - t' + 1)$ th response (and any future responses) to $(ssid_S, a)$ are correctly formed from the point of view of an adversary who wishes

⁶ Though somewhat unusual, the combination of the GapOMDH and DDH assumptions on group \mathbb{G} is not theoretically problematic. It has been proven that the DDH assumption [11] and the GapOMDH assumption [40] each hold for generic groups. Therefore, our security statement is, at a minimum, sound in the Generic Group Model. We also note that there are precedents for making a Gap assumption alongside the DDH assumption on the same group, e.g. [44].

to interpolate a^k . The reduction embeds the OMDH challenge group elements in the H_1 responses, and uses the H_3 random oracle queries as an opportunity to intercept completed evaluations. The “Gap” DDH oracle is used to verify whether or not an H_3 query represents a correct evaluation. If the adversary ever exceeds their $\mathcal{F}_{\text{TOPRF}}$ -allowed number of evaluations, then the reduction exceeds its OMDH-allowed number of k exponentiations and thereby wins the GapOMDH game.

In this GapOMDH reduction, the first $t - t'$ uncorrupted server evaluation requests for any $(ssid_S, a)$ return random group elements, rather than $a^{k_i} \cdot H_2(ssid_S, a)^{z_i}$ for consistent (secret) (k_i, z_i) as in the real world. This is the only difference between the adversary’s views in the reduction and in the real world. We use a series of hybrids to prove that it is not detectable.

Our proof first picks any arbitrary set of $t - t'$ uncorrupted servers. In a series of incremental hybrids, we replace the blinding factor $H_2(ssid_S, a)^{z_i}$ at each of these servers with a randomly chosen group element (for each $(ssid_S, a)$), one by one. For the uncorrupted servers outside of this set, the blinding factors are computed by interpolation in the exponent between the t' key shares attributed to the corrupted servers, the $t - t'$ randomly chosen blinding factors, and the fact that $\{z_i\}$ are a sharing of 0. By a one-time pad argument, a uniformly random blinding factor creates a uniformly random overall server response. Therefore, once all blinding factors are randomized in this way, the adversary’s view is identical to that in the GapOMDH reduction. At each hybrid, we use a DDH reduction to prove that replacing one more server’s blinding factors with random values is not detectable by the adversary. In sum, then, the adversary’s behavior in the GapOMDH reduction must differ only negligibly from the real world.

The factor of $\sum_{t'=0}^t \binom{n}{t'}$ that appears in the adaptive-case security bound is a consequence of the GapOMDH reduction simply guessing (at the start of execution) which t' servers the adversary will eventually corrupt. As long as n and t are small, this guess will succeed with non-negligible probability. It is not sufficient for the reduction to guess a superset (e.g. a t -size superset) of the servers that will eventually be corrupted. The reduction relies on the fact that adversarial computation of a PRF output without having queried $t - t' + 1$ uncorrupted servers always corresponds to a win in the GapOMDH game. This is not true unless the reduction’s guess set is exactly the same as the actual corrupted set at the moment of this adversarial computation.

3.4 Extension to Threshold Partially Oblivious PRF

Partially Oblivious Pseudorandom Function (POPRF) [27] is a generalization of OPRF where the argument to the PRF is split into two parts, x_{priv} and x_{pub} . Function evaluation is only partially oblivious because the x_{pub} part of the input is visible to both parties, while x_{priv} is visible only to the evaluator and is hidden from the server. Note that if POPRF $F_k(x_{\text{priv}}, x_{\text{pub}})$ is evaluated s.t. x_{pub} is always \perp (or any other constant), then POPRF behaves exactly like a standard OPRF, hence POPRF can be seen as a generalization of OPRF.

The techniques we use above to implement a UC threshold OPRF (tOPRF) extend to a UC threshold POPRF (tPOPRF). In the full version of this paper [33] we define tPOPRF via the UC functionality $\mathcal{F}_{\text{tPOPRF}}$, a generalization of our $\mathcal{F}_{\text{OPRF}}$ functionality in Fig. 2, and we show that this functionality is realized by protocol P3HashTDH, which combines the blinding technique used in our 3HashTDH tOPRF protocol with the natural threshold implementation of the pairing-based (single-server) POPRF protocol of Pythia [27]. Protocol P3HashTDH uses only two flows, just like 3HashTDH, and it implements the PRF of Pythia, i.e. $F_k(x_{\text{priv}}, x_{\text{pub}}) = H_3(x_{\text{priv}}, x_{\text{pub}}, e(H_1(x_{\text{priv}}), H'_1(x_{\text{pub}}))^k)$.

We note that [41, 62] showed alternative POPRF constructions that do not rely on pairings. Both of these constructions should have efficient threshold implementations which realize functionality $\mathcal{F}_{\text{tPOPRF}}$ without bilinear maps. The threshold version of the POPRF of [62] would require additional rounds of communication, while POPRF of [41] is a generic construction from any OPRF, and its threshold implementation can be instantiated using our 2-flows tOPRF protocol 3HashTDH. However, the disadvantage of the latter tPOPRF, just like the POPRF of [41], is that it is efficient only for small groups of servers and it offers no verifiability.

4 Augmented Threshold PAKE Model

Figures 4, 5, and 6 are $\mathcal{F}_{\text{atPAKE}}$, the UC functionality for *augmented threshold PAKE (atPAKE)*. As explained in the introduction, this functionality is flexible in that it models target servers and auxiliary servers separately. The target servers are the entities that ultimately wish to establish keys with password-authenticated users. The auxiliary servers distribute the secret information in such a way that it requires the participation of $t+1$ of them for a user to establish a session with a target server. If this separation of responsibilities is undesired, atPAKE can simply be instantiated such that the auxiliary and target server lists partially or wholly overlap.

The **shadowed text** in $\mathcal{F}_{\text{atPAKE}}$ corresponds to relaxations introduced by [43] to the (non-threshold) $\mathcal{F}_{\text{saPAKE}}$ model. The OPAQUE protocol [43] realizes $\mathcal{F}_{\text{saPAKE}}$ only with these slight relaxations, but, as argued in [43], the relaxations do not reduce the functionality’s practical security properties in any significant way. Since our $\mathcal{F}_{\text{atPAKE}}$ is a threshold generalization of [43]’s $\mathcal{F}_{\text{saPAKE}}$, we inherit the same relaxations.

Initialization. A user U may initialize with a group of auxiliary servers and a group of target servers on password pw , represented by sid_A and sid_T respectively, using a `userinit` call to the functionality. Similarly, a server (auxiliary or target) may initialize with a user U using an `auxinit` or `targetinit` call. The (ideal) adversary finishes initialization for an auxiliary server $\mathbf{S}_{\text{sid}_A}[i]$ by sending `finishauxiliaryinit`, which establishes the server’s file record; the file record is `COMPR` (i.e., the adversary knows its content) if the server is corrupt. Similarly, a `finishtargetinit` call finishes initialization for a target server $\mathbf{S}_{\text{sid}_T}[j]$ and establishes a corresponding record, which is `COMPR` if that server is corrupt. However,

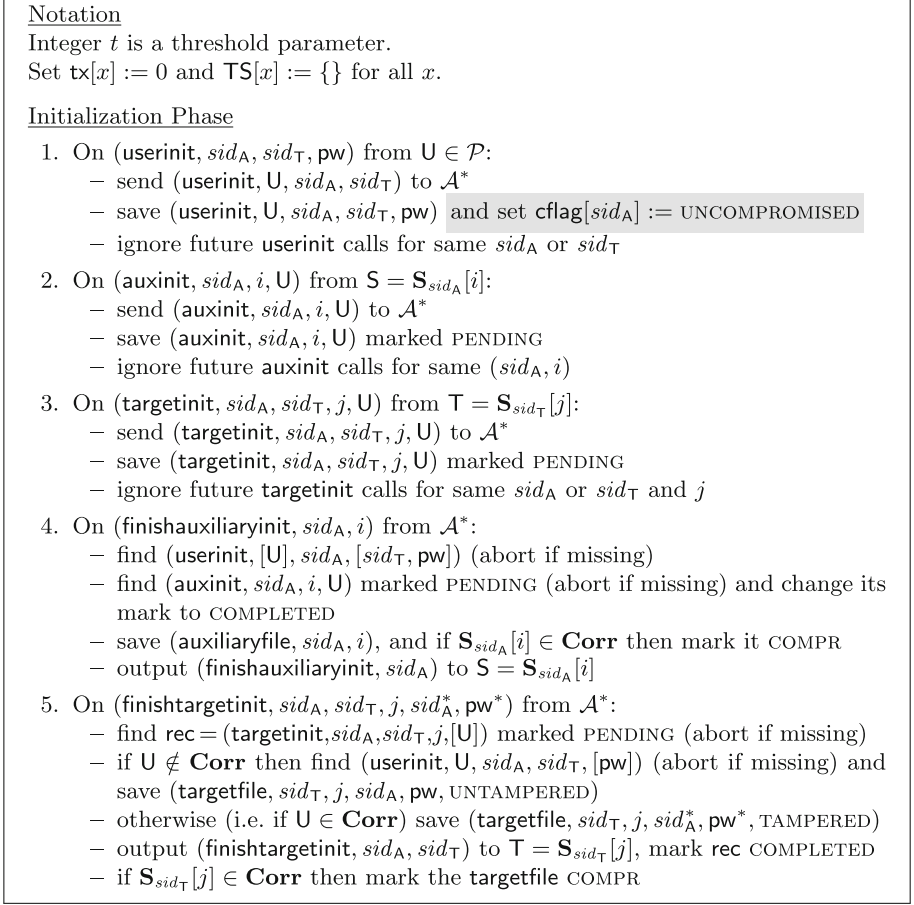


Fig. 4. $\mathcal{F}_{\text{atPAKE}}$: atPAKE functionality (1): Initialization Phase.

a target server's record might additionally be **TAMPERED** if the user it communicates with is corrupt, and in this case we allow the adversary to change the password in the target server's file from pw to some pw^* of the adversary's choice; the adversary can also overwrite the auxiliary server instance sid_A with which the target server's file is associated. As will be apparent in the other sections of the functionality, this **TAMPERED** case models the intuitive notion that many atPAKE security properties are lost if the original initializing user is dishonest.

Corruption, File Compromise, and Offline Password Tests. The adversary may corrupt any party (by sending **corrupt**) or compromise any server and steal its file without corrupting it (by sending **stealauxiliaryfile** or **stealtargetfile**); either way, the corresponding server's file will become **COMPR**.

Obtaining a target server's file allows the adversary to run an offline dictionary attack on it, which is modeled by the **offlinetestpwd** call in which the adver-

Party Corruption, File Compromise, Offline Password Tests	
6.	On (corrupt, P) from \mathcal{A}^* (permitted by \mathcal{Z}), set $\mathbf{Corr} := \mathbf{Corr} \cup \{P\}$ – if \exists (auxiliaryfile, sid_A, i) for $\mathbf{S}_{sid_A}[i] = P$ mark it COMP – if \exists (targetfile, $sid_T, j, sid_A, [pw, tflag]$) for $\mathbf{S}_{sid_T}[j] = P$ mark it COMP
7.	On (steal auxiliaryfile, sid_A, i) from \mathcal{A}^* (permitted by \mathcal{Z}): – if \exists (auxiliaryfile, sid_A, i) mark it COMP
8.	On (steal targetfile, sid_T, j) from \mathcal{A}^* (permitted by \mathcal{Z}): – if \exists (targetfile, $sid_T, j, [sid_A, pw, tflag]$) mark it COMP
9.	On (offlinetestpwd, $sid_A, i, ssid_A, sid_T, j, pw^*$) from \mathcal{A}^* : – if $\text{tx}[sid_A, i, ssid_A] > 0$ add i to $\mathbf{TS}[sid_A, pw^*, ssid_A]$, set $\text{tx}[sid_A, i, ssid_A]--$ – retrieve $\text{rec} = (\text{targetfile}, sid_T, j, sid_A, [pw, tflag])$ (abort if not found) – if $ \mathbf{TS}[sid_A, pw^*, ssid_A] \geq t+1$ and rec is marked COMP then return “correct guess” to \mathcal{A}^* and set $\text{cflag}[sid_A] := \text{COMP}$ if $pw^* = pw$, else return “wrong guess” to \mathcal{A}^*
Authentication Phase (I): Session Initialization, Passive Transmission	
10.	On (usersession, $sid_A, sid_T, j, ssid, pw'$) from $U' \in \mathcal{P}$: – send (usersession, $U', sid_A, sid_T, j, ssid$) to \mathcal{A}^* – save (session, $U', \mathbf{S}_{sid_T}[j], sid_A, sid_T, j, ssid, pw'$) marked PRELIM – ignore future usersession calls for same $ssid$
11.	On (auxsession, $sid_A, i, ssid_A$) from $\mathbf{S} = \mathbf{S}_{sid_A}[i]$ or \mathcal{A}^* : – retrieve (auxiliaryfile, sid_A, i) (abort if record not found) – if sender is \mathcal{A}^* then abort unless the retrieved record is marked COMP – send (auxsession, $sid_A, i, ssid_A$) to \mathcal{A}^* – set $\text{tx}[sid_A, i, ssid_A]++$
12.	On (targetsession, $sid_T, j, U', ssid$) from $\mathbf{S} = \mathbf{S}_{sid_T}[j]$: – retrieve (targetfile, $sid_T, j, [sid_A, pw, tflag]$) (abort if record not found) – send (targetsession, $sid_T, j, U', ssid$) to \mathcal{A}^* – save (session, $\mathbf{S}, U', sid_A, sid_T, j, ssid, pw$) marked FRESH – ignore future targetsession calls for same $ssid$
13.	On (auxproceed, $U', ssid, sid_A^*, ssid_A, \mathbf{C}$) s.t. $ \mathbf{C} = t+1$ from \mathcal{A}^* : – retrieve $\text{rec} = (\text{session}, U', \mathbf{S}, [sid_A, sid_T, j], ssid, [pw'])$ marked PRELIM (abort if record not found) – reset field sid_A in record rec to sid_A^* – abort if $\exists_{i \in \mathbf{C}} \text{tx}[sid_A, i, ssid_A] = 0$, else $\forall_{i \in \mathbf{C}}$ set $\text{tx}[sid_A, i, ssid_A]--$ – change rec 's mark to FRESH
14.	On (testabort, $U', sid_T, j, ssid$) from \mathcal{A}^* : – retrieve $\text{rec} = (\text{session}, U', \mathbf{S}_{sid_T}[j], [sid_A], sid_T, j, ssid, [pw'])$ marked FRESH and (targetfile, $sid_T, j, [sid_A, pw, tflag]$) (abort if either not found) – if $(sid_A', pw') = (sid_A, pw)$ send “success” to \mathcal{A}^* ; else mark rec COMPLETED, send “fail” to \mathcal{A}^* , and output (abort, $ssid$) to U'

Fig. 5. $\mathcal{F}_{\text{atPAKE}}$: atPAKE functionality (2): Compromises, Authentication (I).

sary specifies a password guess pw^* . Though offline with regard to the target server, the adversary still requires the participation of $t+1$ auxiliary servers to perform this attack (some or all of them may be compromised, in which case the adversary can also emulate their participation offline via auxsession, explained

Authentication Phase (II): Active Attacks, Session Termination	
15.	On (auxactive, U' , $ssid$) from \mathcal{A}^* : – retrieve (session, U' , S , [sid_A , sid_T , j], $ssid$, [pw']) marked PRELIM and mark it COUNTERFEIT (abort if record not found)
16.	On (interrupt, sid_T , j , $ssid$) from \mathcal{A}^* : – retrieve (session, $S_{sid_T}[j]$, [U' , sid_A], sid_T , j , $ssid$, [pw]) marked FRESH, mark it INTERRUPTED and set $dPT[ssid] := 1$.
17.	On (testpwd, P , $ssid$, pw^*) from \mathcal{A}^* : – retrieve $rec =$ (session, P , [P' , sid_A , sid_T , j], $ssid$, [pw]) (abort if not found) – if $dPT[ssid] = 1$, then set $dPT[ssid] := 0$; else if rec is not marked FRESH or COUNTERFEIT, abort – if $pw^* = pw$ and any of the following conditions hold: (a) $\exists ssid_A$ s.t. $ TS[sid_A, pw^*, ssid_A] \geq t+1$ (b) or rec marked COUNTERFEIT (c) or $P = S_{sid_T}[j]$ and \exists record (targetfile, sid_T , j , sid_A , pw , TAMPERED) then mark rec as COMPR, send “correct guess” to \mathcal{A}^* , and (if $P = S_{sid_T}[j]$) set $cflag[sid_A] := COMPR$; else mark rec as INTERRUPTED and send “wrong guess” to \mathcal{A}^*
18.	On (impersonate, sid_T , j , $ssid$) from \mathcal{A}^* : – retrieve $rec =$ (session, U' , $S_{sid_T}[j]$, [sid_A], sid_T , j , $ssid$, [pw]) marked FRESH (abort if not found) – if \exists record (targetfile, sid_T , j , sid_A , pw , [tflag']) marked COMPR then mark rec as COMPR and send “correct guess” to \mathcal{A}^* ; else mark rec as INTERRUPTED and send “wrong guess” to \mathcal{A}^*
19.	On (newkey, P , $ssid$, K^*) from \mathcal{A}^* : – retrieve $rec =$ (session, P , [P' , sid_A , sid_T , j], $ssid$, [pw]) not marked PRELIM or COMPLETED (abort if record not found) and do: • if rec is marked COMPR, then set $K \leftarrow K^*$ • if $P = S_{sid_T}[j]$, rec is marked INTERRUPTED, and ($cflag[sid_A] = COMPR$ or \exists record (targetfile, sid_T , j , sid_A , pw , TAMPERED)), then set $K \leftarrow K^*$ • if rec is FRESH and $\exists rec' =$ (session, P' , P , sid_A , sid_T , j , $ssid$, pw) s.t. P' received (newkey, $ssid$, K') when rec' was FRESH, then set $K \leftarrow K'$ • else pick $K \leftarrow_s \{0, 1\}^\tau$ – finally, mark rec as COMPLETED, output (newkey, $ssid$, K) to P

Fig. 6. \mathcal{F}_{atPAKE} : atPAKE functionality (3): Authentication (II).

below). In each `offlinetestpwd` call, the adversary can specify an auxiliary server i with which it wishes to evaluate pw^* . Only once $t + 1$ auxiliary servers have participated in the evaluation of pw^* can the adversary test the password guess pw^* against the compromised target server file and learn whether or not it is correct.

Authentication. In the authentication phase, a user U' may start an online session with a target server $T = S_{sid_T}[j]$ using a `usersession` call (which specifies a password pw'); this call also implicitly defines the auxiliary servers by specifying

sid_A . This establishes a session record for U' marked PRELIM. Similarly, a target server T may start a session with a user U' using a `targetsession` call; since the target server's password is included in its file record, it is not explicitly specified in the `targetsession` message. This establishes a session record for T marked FRESH. Next, an auxiliary server $S_i = S_{sid_A}[i]$ may choose to participate via a `auxsession` call, which also increments its ticket count $tx[sid_A, i, ssid_A]$. If the auxiliary server's file is COMPR, then the adversary can call `auxsession` on its behalf and thereby print tickets for the server at will. In order to progress a user session from PRELIM to FRESH, the adversary must use an `auxproceed` call to connect the user with a set C of $t + 1$ auxiliary servers running on identifier sid_A . Those auxiliary servers must have all agreed to participate in an evaluation, which is tracked via the ticket mechanism.

Once both U' and T sessions are FRESH, the adversary lets a party output a session key using the `newkey` call. If both sides use the same `pw` and the same subsession identifier `ssid`, then they will receive the same key. Otherwise, they will output independently random keys for the session. Figure 7 diagrams the state transitions that user and server session records can move through (including those corresponding to attack scenarios).

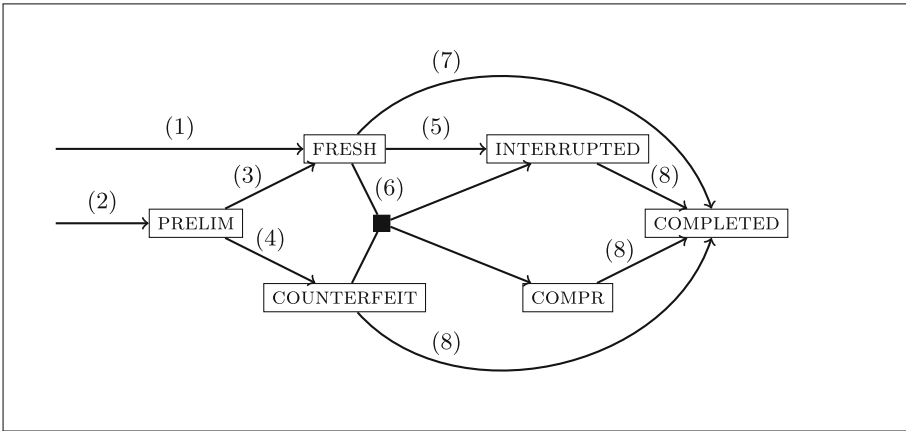


Fig. 7. \mathcal{F}_{atPAKE} session record state diagram. (1) is `targetsession`; (2) is `usersession`; (3) is `auxproceed`; (4) is `auxactive`; (5) is `interrupt` (only for server sessions); (6) is `impersonate` (only for FRESH user sessions) or `testpwd`; (7) is `testabort` (only for user sessions) or `newkey`; (8) is `newkey`.

Passive Attacks. The adversary has two “passive” attack avenues that correspond to simply transmitting messages between parties. With `auxproceed`, the adversary might choose to connect the user with a different auxiliary session sid_A^* than expected; in that case the user will not be able to successfully authenticate to the intended target server (unless that server was dishonestly initialized

with the same spurious sid_A^*). With the `testabort` call, the adversary can connect a FRESH user session to any target server and observe whether or not they successfully authenticate.

Active Attacks. For active session attacks, if the user U' 's session record is PRELIM (i.e. it has not completed its communication with the auxiliary servers), the adversary can run the auxiliary servers' algorithms on its own and communicate with U' . In this situation, modeled by the `auxactive` call, the adversary effectively controls all auxiliary servers that communicate with U' , so we mark U' 's session record COUNTERFEIT. Such a session can never successfully authenticate to a target server, but it is vulnerable to an online password-guessing attack.

Using `testpwd`, the adversary can perform an active session attack with some password guess pw^* . There are three possible cases:

- (a) the adversary has evaluated pw^* with $t + 1$ auxiliary servers (using `offlinetestpwd`);
- (b) the adversary is attacking a user session that is COUNTERFEIT;
- (c) the adversary is attacking a server that was dishonestly initialized (i.e. its file is TAMPERED).

If the password guess is successful, then the attacked session is marked COMPR. Otherwise, it is marked INTERRUPTED, indicating that it is no longer possible for the session to successfully complete. In either case, the adversary learns whether or not the password guess was correct. After stealing a target server's file, the adversary can also compromise user sessions meant to connect with that server via the `impersonate` call. If the user's password does not match the one in the saved file, then this `impersonate` attack will fail and the user's session will become INTERRUPTED.

Once a session is COMPR, the adversary has done a successful attack, so the adversary is able to choose that session's output key in `newkey`. If both the session and its countersession are FRESH, this models an unattacked pair, so the two parties output the same random key (if their passwords and sid_{AS} match). In all other cases (i.e. COUNTERFEIT and INTERRUPTED), the functionality samples an independent random key for the session.

Comparison to Game-Based tPAKE. As a sanity check, we verify that $\mathcal{F}_{\text{atPAKE}}$ is at least as strong as the game-based tPAKE definition of MacKenzie, Shrimpton, and Jakobsson [52]. In the full version of the paper we show a proof that any protocol realizing $\mathcal{F}_{\text{atPAKE}}$ (in the case that the auxiliary and target server lists are identical) is also secure under that notion.

5 Augmented Threshold PAKE Construction

Figure 8 shows protocol $\Pi_{\text{tOPRF-atPAKE}}$, our main atPAKE construction which is a generic composition of UC tOPRF and UC (strong) aPAKE. In Fig. 8 we show this protocol in the hybrid model assuming functionalities $\mathcal{F}_{\text{tOPRF}}$ (shown in Fig. 2) and $\mathcal{F}_{\text{saPAKE}}$ which model respectively UC tOPRF and UC (strong)

Initialization

1. On input $(\text{atpake.userinit}, sid_A, sid_T, pw)$, user U does:
 - send $(\text{toprf.init}, sid_A, pw)$ to $\mathcal{F}_{\text{tOPRF}}$
 - await response $(\text{toprf.initeval}, sid_A, rw)$
 - for every $j \in \{1, \dots, |\mathbf{S}_{sid_T}|\}$, compute $rw_j : \text{KDF}(rw, \mathbf{S}_{sid_T}[j])$ and send $(\text{channel.send}, (sid_A || sid_T || j), \mathbf{S}_{sid_T}[j], rw_j)$ to $\mathcal{F}_{\text{channel}}$
2. On input $(\text{atpake.auxinit}, sid_A, i, U')$, auxiliary server $\mathbf{S}_{sid_A}[i]$ does:
 - send $(\text{toprf.sinit}, sid_A, i, U')$ to $\mathcal{F}_{\text{tOPRF}}$
 - await response $(\text{toprf.finit}, sid_A, i)$
 - then output $(\text{atpake.finishauxiliaryinit}, sid_A)$
3. On input $(\text{atpake.targetinit}, sid_A, sid_T, j, U')$, target server $\mathbf{S}_{sid_T}[j]$ does:
 - await $(\text{channel.deliver}, (sid_A || sid_T || j), U', rw_j)$ from $\mathcal{F}_{\text{channel}}$
 - send $(\text{sapake.storepwdfile}, (sid_T || j), U', rw_j)$ to $\mathcal{F}_{\text{saPAKE}}$
 - output $(\text{atpake.finishtargetinit}, sid_A, sid_T)$

Authentication

4. On input $(\text{atpake.usersession}, sid_A, sid_T, j, ssid, pw')$, user U' does:
 - send $(\text{toprf.eval}, sid_A, ssid, pw')$ to $\mathcal{F}_{\text{tOPRF}}$
 - await response $(\text{toprf.eval}, sid_A, ssid, rw')$
 - compute $rw'_j := \text{KDF}(rw', \mathbf{S}_{sid_T}[j])$
 - send $(\text{sapake.usrsession}, (sid_T || j), ssid, \mathbf{S}_{sid_T}[j], rw'_j)$ to $\mathcal{F}_{\text{saPAKE}}$
 - upon response $(\text{sapake.newkey}, (sid_T || j), ssid, K)$, output $(\text{atpake.newkey}, ssid, K)$
 - upon response $(\text{sapake.abort}, (sid_T || j), ssid)$, output $(\text{atpake.abort}, ssid)$
5. On input $(\text{atpake.auxsession}, sid_A, i, ssid_A)$, auxiliary server $\mathbf{S}_{sid_A}[i]$ sends $(\text{toprf.sndrcomplete}, sid_A, i, ssid_A)$ to $\mathcal{F}_{\text{tOPRF}}$
6. On input $(\text{atpake.targetsession}, sid_T, j, U', ssid)$, target server $\mathbf{S}_{sid_T}[j]$ does:
 - send $(\text{sapake.svrsession}, (sid_T || j), ssid)$ to $\mathcal{F}_{\text{saPAKE}}$
 - await response $(\text{atpake.newkey}, (sid_T || j), ssid, K)$
 - output $(\text{atpake.newkey}, ssid, K)$

Fig. 8. Protocol $\Pi_{\text{tOPRF-atPAKE}}$ which realizes $\mathcal{F}_{\text{atPAKE}}$ using $\mathcal{F}_{\text{tOPRF}}, \mathcal{F}_{\text{saPAKE}}$

aPAKE, but in an implementation these functionalities will be replaced by sub-protocols that realize them. Protocol $\Pi_{\text{tOPRF-atPAKE}}$ also uses secure channels modeled by functionality $\mathcal{F}_{\text{channel}}$ (shown in Fig. 1), but it uses them only in the initialization. Note that our realization of the threshold OPRF functionality $\mathcal{F}_{\text{tOPRF}}$, i.e. protocol 3HashTDH of Sect. 3.2, also relies on secure channels in the initialization.

In protocol $\Pi_{\text{tOPRF-atPAKE}}$, authentication between a user and a target server T , indexed as the j -th server in the target server list \mathbf{S}_{sid_T} , proceeds in two steps. First, the user interacts with $t + 1$ tOPRF servers, i.e. the auxiliary servers, in order to convert their password guess pw' into a “hardened” password $rw' = F_k(pw')$. Then, the user uses a pseudorandom function to derive

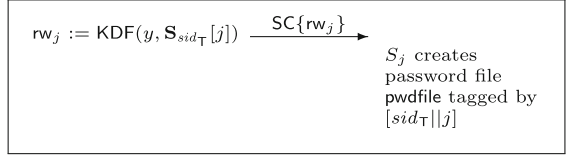
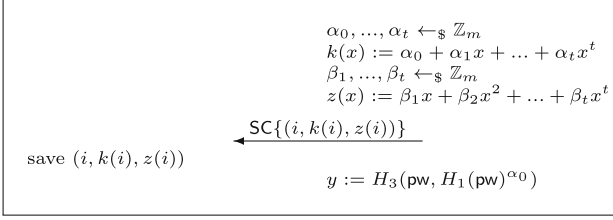
Notation As in Section 2, τ is a security parameter, $\text{KDF} : \{0, 1\}^\tau \times \{0, 1\}^* \rightarrow \{0, 1\}^\tau$ is a PRF, \mathbb{G} is a group of prime order m . H_1, H_2, H_3 are hash functions with ranges \mathbb{G}, \mathbb{G} , and $\{0, 1\}^l$ where l is a parameter. SC denotes communication over a secure and authenticated channel. “aPAKE” denotes arbitrary (strong) aPAKE. (In other protocol variants aPAKE can be replaced with TLS-OPAQUE, weak aPAKE, envelope+aKE, or password-over-TLS.)

Initialization

Auxilliary Server S_i on
(sid_A, i, U)

User U on
(sid_A, sid_τ, pw)

Target Server S_j on
(sid_A, sid_τ, j, U)



Authentication

Auxilliary Server S_i on
($sid_A, i, ssid_A$)

User U on
($sid_A, sid_\tau, j, ssid, pw$)

Target Server S_j on
($sid_\tau, j, U, ssid$)

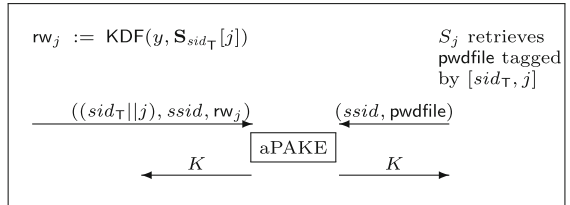
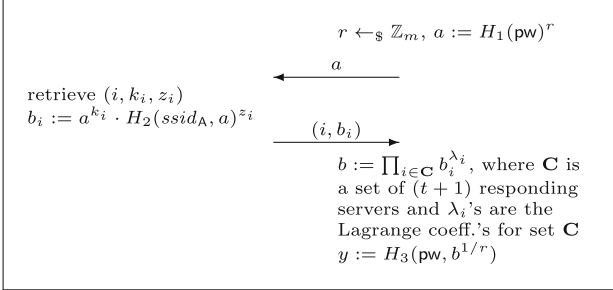


Fig. 9. Concrete instantiation of $\Pi_{\text{tOPRF-atPAKE}}$ using arbitrary (strong) aPAKE

a \mathbb{T} -specific password $\text{rw}'_j = \text{KDF}(\text{rw}', \mathbb{T})$, and uses rw'_j as the password in an underlying (strong) aPAKE instance between the user and the target server.

When registering a new user with password pw , the client must initialize a new tOPRF instance $F_k(\cdot)$ with the auxiliary servers. At the same time the client computes $\text{rw}_j = \text{KDF}(F_k(\text{pw}), \mathbb{T}_j)$ for every target server $\mathbb{T}_j = \mathbf{S}_{\text{sid}_{\mathbb{T}}}[j]$, and sends rw_j to \mathbb{T}_j over a secure channel. Upon receiving rw_j , server \mathbb{T}_j uses it to create a password file for this user.

Theorem 3. *Protocol $\Pi_{\text{tOPRF-atPAKE}}$ realizes functionality $\mathcal{F}_{\text{atPAKE}}$ with parameters t and n in the $(\mathcal{F}_{\text{tOPRF}}, \mathcal{F}_{\text{saPAKE}}, \mathcal{F}_{\text{channel}})$ -hybrid model.*

Specifically, for any efficient adversary \mathcal{A} against protocol $\Pi_{\text{tOPRF-atPAKE}}$, there exists a simulator SIM such that no efficient environment \mathcal{Z} can distinguish the view of \mathcal{A} interacting with the real $\Pi_{\text{tOPRF-atPAKE}}$ protocol and the view of SIM interacting with the ideal functionality $\mathcal{F}_{\text{atPAKE}}$ with advantage better than $(q_{\mathbb{T}} \cdot q_{\text{eval}}^2 + q_{\text{test}} + q_{\mathbb{T}}^ \cdot q_{\text{eval}}) / 2^\tau$ where $q_{\mathbb{T}}$ is the number of target server instances, q_{eval} is the number of tOPRF evaluations, q_{test} is the number of online and offline password-guessing attacks against $\mathcal{F}_{\text{saPAKE}}$, $q_{\mathbb{T}}^*$ is the number of dishonestly initialized target server instances, and security parameter τ is the tOPRF output length.*

Proof of Theorem 3 is given in the full version of the paper.

Concrete Instantiation. In Fig. 9 we show a concrete instantiation of protocol $\Pi_{\text{tOPRF-atPAKE}}$, with $\mathcal{F}_{\text{tOPRF}}$ realized with the 3HashTDH protocol of Sect. 3.2. The user and the target server interact via an arbitrary (strong) aPAKE, which can be realized with any realization of $\mathcal{F}_{\text{saPAKE}}$, including OPAQUE [43], OPAQUE' [34, 43, 56], TLS-OPAQUE [36, 60], or other aPAKE constructions [13, 53]. As discussed in Sect. 6, the (strong) aPAKE can be replaced with weaker building blocks, including weak aPAKE [31], the “envelope+AKE” building block used within OPAQUE, or even password-over-TLS, although the resulting protocol could realize modified (and sometimes weakened) versions of the atPAKE functionality $\mathcal{F}_{\text{atPAKE}}$.

6 Protocol Variants and Extensions

We considered several possible variants of the tOPRF+aPAKE construction of atPAKE shown in Sect. 5. Indeed, both the UC tOPRF subprotocol and the (strong) aPAKE protocol can be substituted by other building blocks, and the resulting protocols implement variants of the atPAKE functionality $\mathcal{F}_{\text{atPAKE}}$. We summarize the security properties of the protocol variants we considered in Table 1 below. Table entries marked with a special symbol (*) are verified formally either here or in the full version of the paper [33]. All the other table entries are not formally verified, but these are our hypotheses based on extrapolating the formally verified cases.

Implementing atPAKE with Variants of tOPRF. The columns of the table include three variants of the threshold OPRF (tOPRF) protocol, namely tOPRF

itself, the threshold *Partial* OPRF (tPOPRF) (see Sect. 3.4), and the *augmented* Password-Protected Secret-Sharing (aPPSS) [26], an extension of PPSS [6] to security up to offline dictionary attack upon compromise of all servers. Recall that PPSS is a protocol which secret-shares a secret s among n servers and protects it under password pw , s.t. no t parties can learn any information on s , and $t + 1$ parties suffice to reconstruct s but only if the reconstruction protocol client enters the same password pw which was used to initialize this secret-sharing.⁷

These three protocol variants have the following security characteristics: Replacing tOPRF with tPOPRF creates a very mild and essentially negligible difference in the atPAKE security model, denoted $\mathcal{F}_{\text{atPAKE}'}$ (see [33] for details). However, replacing t(P)OPRF with aPPSS changes the resulting atPAKE notion to a variant of UC atPAKE notion we denote $\mathcal{F}_{\text{atPAKE}(\text{weak})}$, which is weaker than $\mathcal{F}_{\text{atPAKE}}$ in the following sense: In the latter model an offline dictionary attack (ODA) is enabled only if the adversary corrupts $t + 1$ of n auxiliary servers S_1, \dots, S_n and the target server T , whereas in $\mathcal{F}_{\text{atPAKE}(\text{weak})}$ the ODA requires corruption of $t + 1$ auxiliary servers, but it can be done without corrupting a target server. However, note that one can implement the security contract of $\mathcal{F}_{\text{atPAKE}}$ using $\mathcal{F}_{\text{atPAKE}(\text{weak})}$ if the atPAKE scheme involves a single target server: If the target server T is part of the auxiliary server group, and it holds a “blocking” set of shares, then corruption of a sufficient number of (virtual) auxiliary servers becomes possible only if one corrupts $t + 1$ (real) auxiliary server and the target server T .

Table 1. Summary of security properties of atPAKE implementation variants

U-to-T subprotocol	tOPRF	tPOPRF	aPPSS
(strong) aPAKE	$\mathcal{F}_{\text{atPAKE}}^{(*)}$	$\mathcal{F}_{\text{atPAKE}'}^{(*)}$	$\mathcal{F}_{\text{atPAKE}(\text{weak})}^{(*)}$
TLS-OPAQUE	$\mathcal{F}_{\text{atPAKE-EA}}$	$\mathcal{F}_{\text{atPAKE}'-EA}$	$\mathcal{F}_{\text{atPAKE}(\text{weak})-EA}$
password-over-TLS	$\mathcal{F}_{\text{atPAKE-PKI}}$	$\mathcal{F}_{\text{atPAKE}'-PKI}$	$\mathcal{F}_{\text{atPAKE}(\text{weak})-PKI}$
weak aPAKE	$\mathcal{F}_{\text{atPAKE}(\text{medium})}^{(*)}$	$\mathcal{F}_{\text{atPAKE}'(\text{medium})}$	$\mathcal{F}_{\text{atPAKE}(\text{weak})}$
AKE	N/A	N/A	$\mathcal{F}_{\text{atPAKE}(\text{weak})}$

Implementing atPAKE with Variants of (Strong) aPAKE. Table rows include five variants of the U-to-T authentication protocol, i.e. the way user U uses rw retrieved using its password to authenticate to the target server T . In the protocol analyzed in Sect. 5 this is handled by (strong) aPAKE. The same holds for the variants examined in the full version of the paper [33], where

⁷ As shown in [40], tOPRF implies PPSS: One simply uses an authenticated encryption to encrypt secret s under $\text{rw} = F_k(\text{pw})$, where F_k is a tOPRF implemented by the servers. We believe that this implements UC *augmented* PPSS (aPPSS) [26], but we leave formal verification of this to future work.

tOPRF is replaced by tPOPRF and aPPSS. However, one can consider replacing the (strong) aPAKE sub-protocol with TLS-OPAQUE, i.e. the *Exported Authenticator* (EA) extension of TLS which implements augmented password authentication over existing TLS connection, rather than creating new password-authenticated session key [36,60]. The mechanics of these “EA” extensions of our $\mathcal{F}_{\text{atPAKE}}$ variants will be similar as in [36], and their security properties should be the same as in $\mathcal{F}_{\text{atPAKE}}/\mathcal{F}_{\text{atPAKE}'}/\mathcal{F}_{\text{atPAKE}(\text{weak})}$ except the final U-T authentication would pertain to a pre-established secure channel between these two parties.

Another possible variant considers U-T interaction implemented as “password-over-TLS”. Such “PKI” extensions of our $\mathcal{F}_{\text{atPAKE}}$ variants would have weaker security: First, U’s login sessions need to include T’s identity as input. Second, if this identity is wrong, i.e. U fails to authenticate the proper T counterparty, or if it is right but T is compromised, then (1) the adversary learns rw_T and can authenticate to T as U, and (2) the adversary can stage ODA if $t + 1$ auxiliary servers are corrupted. (In other words, PKI error in U authenticating T has the same consequences regarding ODA attack as corruption of T.)

The final two possibilities include replacing (strong) aPAKE with weak aPAKE of [31] or with pfs-AKE, i.e. AKE with perfect forward secrecy, usually achieved by key confirmation flows. Using the first option allows for atPAKE with lower round complexity (only 3 protocol flows) if weak aPAKE is implemented with a 2-flow protocol of [30]. However, the resulting functionality is slightly weaker in the case of $\mathcal{F}_{\text{atPAKE}}$ and $\mathcal{F}_{\text{atPAKE}'}$, denoted resp. $\mathcal{F}_{\text{atPAKE}(\text{medium})}$ and $\mathcal{F}_{\text{atPAKE}'(\text{medium})}$. The weakening is that after compromise of $t + 1$ auxiliary servers the attacker can precompute the ODA before the compromise of a target server. This forms a mid-point between $\mathcal{F}_{\text{atPAKE}}$, where ODA can start only at compromise of $t + 1$ auxiliary servers and a target server, and $\mathcal{F}_{\text{atPAKE}(\text{weak})}$ where it can start at compromise of just the auxiliary servers. (We include this protocol variant in the full version of the paper [33].) Using pfs-AKE can further lower the U-T subprotocol cost, but it can be done only with aPPSS, because U needs to reconstruct structured data, not a pseudorandom string, to run AKE, namely its own private key and server T’s public key. Using either weak aPAKE or pfs-AKE option the aPPSS-based protocol should achieve the same atPAKE notion variant $\mathcal{F}_{\text{atPAKE}(\text{weak})}$, but we recommend that the last option, i.e. aPPSS and pfs-AKE, should be formally verified before anyone implements it.

Proactive Security. In our tOPRF and tPOPRF protocols, the only state held by each server is two Shamir secret shares, one of the PRF key and one of zero. Therefore, these protocols can be naturally extended to support proactive key refresh by using standard techniques for proactively refreshing a secret sharing. For example, [35] presents a scheme where all share-holders broadcast verifiable secret sharings of zero, which are then added to the secret sharing to be refreshed. After carrying out this refresh procedure, the share-holders hold a fresh sharing of their original (unchanged) secret value. If the servers in our t(P)OPRF were to adopt this behavior, it would effectively partition their evaluation tickets into “epochs” separated by key-refresh events. To successfully evaluate, one would

not only need the participation of $t + 1$ servers, but the participation of $t + 1$ servers *within the time of one epoch*. Similarly, the adversary would gain nothing by stealing server files across multiple epochs; only by stealing $t + 1$ servers' data within the time of one epoch could the adversary unlock the power to perform entirely offline evaluations.

Our atPAKE constructions that are based on t(P)OPRF inherit these same proactive security properties. We note that the tOPRF-based atPAKE protocol requires auxiliary servers to hold a separate tOPRF instance for each user, whereas the tPOPRF-based construction reuses a single tPOPRF instance across all users. Since it would likely be impractical for servers to proactively refresh a separate key sharing for every registered user, the tPOPRF-based atPAKE has an additional benefit: Besides reducing each server's storage load, it would also make proactive key refresh much more practical.

References

1. Facebook stored hundreds of millions of passwords in plain text, <https://www.theverge.com/2019/3/21/18275837/facebook-plain-text-password-storage-hundreds-millions-users>. 2019.
2. Google stored some passwords in plain text for fourteen years, <https://www.theverge.com/2019/5/21/18634842/google-passwords-plain-text-g-suite-fourteen-years>. 2019.
3. Michel Abdalla, Olivier Chevassut, Pierre-Alain Fouque, and David Pointcheval. A simple threshold authenticated key exchange from short secrets. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 566–584. Springer, Berlin, Heidelberg, December 2005.
4. Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 65–84. Springer, Berlin, Heidelberg, January 2005.
5. Martin R. Albrecht, Alex Davidson, Amit Deo, and Nigel P. Smart. Round-optimal verifiable oblivious pseudorandom functions from ideal lattices. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 261–289. Springer, Cham, May 2021.
6. Ali Bagherzandi, Stanislaw Jarecki, Nitesh Saxena, and Yanbin Lu. Password-protected secret sharing. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM CCS 2011*, pages 433–444. ACM Press, October 2011.
7. Carsten Baum, Tore Kasper Frederiksen, Julia Hesse, Anja Lehmann, and Avishay Yanai. Pesto: Proactively secure distributed single sign-on, or how to trust a hacked server. In *Proceedings - 5th IEEE European Symposium on Security and Privacy, Euro S and P 2020*, pages 587–606. IEEE, 2020. 2020 IEEE European Symposium on Security and Privacy (EuroS&P) ; Conference date: 07-09-2020 Through 11-09-2020.
8. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Berlin, Heidelberg, May 2000.

9. Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10. ACM Press, May 1988.
10. Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Mitigating server breaches in password-based authentication: Secure and efficient solutions. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 3–18. Springer, Cham, February / March 2016.
11. Dan Boneh. The decision diffie-hellman problem. Stanford Cryptography Group webpage, 1998. <https://crypto.stanford.edu/SIMdabo/pubs/papers/DDH.pdf>
12. D. Bourdrez, H. Krawczyk, K. Lewi, and C. Wood. The OPAQUE Asymmetric PAKE Protocol, draft-irtf-cfrg-opaque, <https://tools.ietf.org/id/draft-irtf-cfrg-opaque>, July 2022.
13. Tatiana Bradley, Stanislaw Jarecki, and Jiayu Xu. Strong asymmetric PAKE based on trapdoor CKEM. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 798–825. Springer, Cham, August 2019.
14. John G. Brainard, Ari Juels, Burt Kaliski, and Michael Szydlo. A new two-server approach for authentication with short secrets. In *USENIX Security 2003*. USENIX Association, August 2003.
15. Jan Camenisch, Robert R. Enderlein, and Gregory Neven. Two-server password-authenticated secret sharing UC-secure against transient corruptions. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 283–307. Springer, Berlin, Heidelberg, March / April 2015.
16. Jan Camenisch, Anja Lehmann, Anna Lysyanskaya, and Gregory Neven. Memento: How to reconstruct your secrets from a single password in a hostile environment. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 256–275. Springer, Berlin, Heidelberg, August 2014.
17. Jan Camenisch, Anja Lehmann, and Gregory Neven. Optimal distributed password verification. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 182–194. ACM Press, October 2015.
18. Jan Camenisch, Anna Lysyanskaya, and Gregory Neven. Practical yet universally composable two-server password-authenticated secret sharing. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 2012*, pages 525–536. ACM Press, October 2012.
19. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.
20. Ran Canetti and Shafi Goldwasser. An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 90–106. Springer, Berlin, Heidelberg, May 1999.
21. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, Berlin, Heidelberg, May 2005.
22. Sílvia Casacuberta, Julia Hesse, and Anja Lehmann. Sok: Oblivious pseudorandom functions. Cryptology ePrint Archive, Paper 2022/302, 2022. <https://eprint.iacr.org/2022/302>.
23. Poulami Das, Julia Hesse, and Anja Lehmann. Dpase: Distributed password-authenticated symmetric encryption. Cryptology ePrint Archive, Paper 2020/1443, 2020. <https://eprint.iacr.org/2020/1443>.

24. Sourav Das and Ling Ren. Adaptively secure BLS threshold signatures from DDH and co-CDH. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 251–284. Springer, Cham, August 2024.
25. Mario Di Raimondo and Rosario Gennaro. Provably secure threshold password-authenticated key exchange. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 507–523. Springer, Berlin, Heidelberg, May 2003.
26. Stefan Dziembowski, Stanislaw Jarecki, Pawel Kedzior, Hugo Krawczyk, Nam Ngo, and Jiayu Xu. Password-protected threshold signatures. IACR Cryptology ePrint Archive, 2024.
27. Adam Everspaugh, Rahul Chatterjee, Samuel Scott, Ari Juels, and Thomas Ristenpart. The pythia PRF service. In Jaeyeon Jung and Thorsten Holz, editors, *USENIX Security 2015*, pages 547–562. USENIX Association, August 2015.
28. Warwick Ford and Burton S. Kaliski Jr. Server-assisted generation of a strong secret from a password. In *9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2000)*, pages 176–180, Gaithersburg, MD, USA, June 4–16, 2000. IEEE Computer Society.
29. Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 303–324. Springer, Berlin, Heidelberg, February 2005.
30. Bruno Freitas Dos Santos, Yanqi Gu, Stanislaw Jarecki, and Hugo Krawczyk. Asymmetric pake with low computation and communication. In *EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2022.
31. Craig Gentry, Philip D. Mackenzie, and Zulfikar Ramzan. Password authenticated key exchange using hidden smooth subgroups. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 2005*, pages 299–309. ACM Press, November 2005.
32. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.
33. Yanqi Gu, Stanislaw Jarecki, Pawel Kedzior, Phillip Nazarian, and Jiayu Xu. Threshold pake with security against compromise of all servers. Cryptology ePrint Archive, report 2024/1455, 2024.
34. Yanqi Gu, Stanislaw Jarecki, and Hugo Krawczyk. KHAPE: Asymmetric PAKE from key-hiding key exchange. In *Advances in Cryptology - Crypto 2021*, pages 701–730, 2021. <https://ia.cr/2021/873>.
35. Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 339–352. Springer, Berlin, Heidelberg, August 1995.
36. Julia Hesse, Stanislaw Jarecki, Hugo Krawczyk, and Christopher Wood. Password-authenticated TLS via OPAQUE and post-handshake authentication. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 98–127. Springer, Cham, April 2023.
37. David P. Jablon. Password authentication using multiple servers. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 344–360. Springer, Berlin, Heidelberg, April 2001.
38. Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part II*, volume 8874 of *LNCS*, pages 233–253. Springer, Berlin, Heidelberg, December 2014.

39. Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. Highly-efficient and composable password-protected secret sharing (or: how to protect your bitcoin wallet online). In *IEEE European Symposium on Security and Privacy – EuroS&P 2016*, pages 276–291. IEEE, 2016.
40. Stanislaw Jarecki, Aggelos Kiayias, Hugo Krawczyk, and Jiayu Xu. TOPPSS: Cost-minimal password-protected secret sharing based on threshold OPRF. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *ACNS 17International Conference on Applied Cryptography and Network Security*, volume 10355 of *LNCS*, pages 39–58. Springer, Cham, July 2017.
41. Stanislaw Jarecki, Hugo Krawczyk, and Jason Resch. Threshold partially-oblivious PRFs with applications to key management. Cryptology ePrint Archive, Report 2018/733, 2018.
42. Stanislaw Jarecki, Hugo Krawczyk, Maliheh Shirvanian, and Nitesh Saxena. Device-enhanced password protocols with optimal online-offline protection. In Xiaofeng Chen, XiaoFeng Wang, and Xinyi Huang, editors, *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS 2016, Xi'an, China, May 30 - June 3, 2016*, pages 177–188. ACM, 2016.
43. Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 456–486. Springer, Cham, April / May 2018.
44. Stanislaw Jarecki and Xiaomin Liu. Affiliation-hiding envelope and authentication schemes with efficient support for multiple credentials. In *Automata, Languages and Programming*, pages 715–726, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.
45. Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 577–594. Springer, Berlin, Heidelberg, March 2009.
46. Haimin Jin, Duncan S. Wong, and Yinlong Xu. An efficient password-only two-server authenticated key exchange system. In Sihan Qing, Hideki Imai, and Guilin Wang, editors, *ICICS 07*, volume 4861 of *LNCS*, pages 44–56. Springer, Berlin, Heidelberg, December 2008.
47. Jonathan Katz, Philip D. MacKenzie, Gelareh Taban, and Virgil D. Gligor. Two-server password-only authenticated key exchange. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *ACNS 05International Conference on Applied Cryptography and Network Security*, volume 3531 of *LNCS*, pages 1–16. Springer, Berlin, Heidelberg, June 2005.
48. Franziskus Kiefer and Mark Manulis. Distributed smooth projective hashing and its application to two-server password authenticated key exchange. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *ACNS 14International Conference on Applied Cryptography and Network Security*, volume 8479 of *LNCS*, pages 199–216. Springer, Cham, June 2014.
49. Franziskus Kiefer and Mark Manulis. Universally composable two-server PAKE. In Matt Bishop and Anderson C. A. Nascimento, editors, *ISC 2016*, volume 9866 of *LNCS*, pages 147–166. Springer, Cham, September 2016.
50. Russell W. F. Lai, Christoph Egger, Dominique Schröder, and Sherman S. M. Chow. Phoenix: Rebirth of a cryptographic password-hardening service. In Engin Kirda and Thomas Ristenpart, editors, *USENIX Security 2017*, pages 899–916. USENIX Association, August 2017.

51. Leona Lassak, Annika Hildebrandt, Maximilian Golla, and Blase Ur. “it’s stored, hopefully, on an encrypted server”: Mitigating users’ misconceptions about fido2 biometric webauthn. In *Proc. USENIX Security*, 2021.
52. Philip D. MacKenzie, Thomas Shrimpton, and Markus Jakobsson. Threshold password-authenticated key exchange. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 385–400. Springer, Berlin, Heidelberg, August 2002.
53. Ian McQuoid and Jiayu Xu. An efficient strong asymmetric PAKE compiler instantiable from group actions. 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 VIII*, volume 14445 of *Lecture Notes in Computer Science*, pages 176–207. Springer, 2023.
54. Kentrell Owens, Olabode Anise, Amanda Krauss, and Blase Ur. User perceptions of the usability and security of smartphones as fido2 roaming authenticators. In *SOUPS*, pages 57–76, 2021.
55. Hirak Ray, Flynn Wolf, Ravi Kuber, and Adam J Aviv. Why older adults (don’t) use password managers. In *USENIX*, 2021.
56. Bruno Freitas Dos Santos, Yanqi Gu, Stanislaw Jarecki, and Hugo Krawczyk. Asymmetric PAKE with low computation and communication. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II*, volume 13276 of *Lecture Notes in Computer Science*, pages 127–156. Springer, 2022.
57. Jonas Schneider, Nils Fleischhacker, Dominique Schröder, and Michael Backes. Efficient cryptographic password hardening services from partially oblivious commitments. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1192–1203. ACM Press, October 2016.
58. Maliheh Shirvanian, Stanislaw Jarecki, Hugo Krawczyk, and Nitesh Saxena. SPHINX: A password store that perfectly hides passwords from itself. In Kisung Lee and Ling Liu, editors, *37th IEEE International Conference on Distributed Computing Systems, ICDCS 2017, Atlanta, GA, USA, June 5-8, 2017*, pages 1094–1104. IEEE Computer Society, 2017.
59. Maliheh Shirvanian, Christopher Robert Price, Mohammed Jubur, Nitesh Saxena, Stanislaw Jarecki, and Hugo Krawczyk. A hidden-password online password manager. In Chih-Cheng Hung, Jiman Hong, Alessio Bechini, and Eunjee Song, editors, *SAC ’21: The 36th ACM/SIGAPP Symposium on Applied Computing, Virtual Event, Republic of Korea, March 22-26, 2021*, pages 1683–1686. ACM, 2021.
60. N. Sullivan, H. Krawczyk, O. Friel, and R. Barnes. OPAQUE with TLS 1.3, draft-sullivan-tls-opaque-01, <https://datatracker.ietf.org/doc/html/draft-sullivan-tls-opaque>, February 2021.
61. Michael Szydło and Burton S. Kaliski Jr. Proofs for two-server password authentication. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 227–244. Springer, Berlin, Heidelberg, February 2005.
62. Nirvan Tyagi, Sofía Celi, Thomas Ristenpart, Nick Sullivan, Stefano Tessaro, and Christopher A. Wood. A fast and simple partially oblivious prf, with applications. In *Advances in Cryptology - EUROCRYPT 2022: 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim*,

Norway, May 30 - June 3, 2022, *Proceedings, Part II*, page 674-705, Berlin, Heidelberg, 2022. Springer-Verlag.

63. Nirvan Tyagi, Sofia Celi, Thomas Ristenpart, Nick Sullivan, Stefano Tessaro, and Christopher A. Wood. A fast and simple partially oblivious prf, with applications. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology – EUROCRYPT 2022*, pages 674–705, Cham, 2022. Springer International Publishing.
64. Yanjiang Yang, Robert Deng, and Feng Bao. A practical password-based two-server authentication and key exchange system. *Dependable and Secure Computing, IEEE Transactions on*, 3:105–114, 05 2006.
65. Lin Zhang, Zhenfeng Zhang, and Xuexian Hu. UC-secure two-server password-based authentication protocol and its applications. In Xiaofeng Chen, XiaoFeng Wang, and Xinyi Huang, editors, *ASIACCS 16*, pages 153–164. ACM Press, May / June 2016.



Key Exchange in the Post-snowden Era: Universally Composable Subversion-Resilient PAKE

Suvradip Chakraborty^{1(✉)}, Lorenzo Magliocco², Bernardo Magri^{3,4},
and Daniele Venturi²

¹ VISA Research, Foster City, USA
suvradip1111@gmail.com

² Sapienza University of Rome, Rome, Italy

³ University of Manchester, Manchester, UK

⁴ Primev, Manchester, UK

Abstract. Password-Authenticated Key Exchange (PAKE) allows two parties to establish a common high-entropy secret from a possibly low-entropy pre-shared secret such as a password. In this work, we provide the first PAKE protocol with *subversion resilience* in the framework of universal composability (UC), where the latter roughly means that UC security still holds even if one of the two parties is malicious and the honest party's code has been subverted (in an undetectable manner).

We achieve this result by sanitizing the PAKE protocol from oblivious transfer (OT) due to Canetti *et al.* (PKC'12) via cryptographic reverse firewalls in the UC framework (Chakraborty *et al.*, EUROCRYPT'22). This requires new techniques, which help us uncover new cryptographic primitives with sanitation-friendly properties along the way (such as OT, dual-mode cryptosystems, and signature schemes).

As an additional contribution, we delve deeper in the backbone of communication required in the subversion-resilient UC framework, extending it to the *unauthenticated* setting, in line with the work of Barak *et al.* (CRYPTO'05).

Keywords: PAKE · subversion resilience · universal composability

1 Introduction

Authenticated Key Exchange (AKE) allows two parties to generate a shared high-entropy secret and mutually authenticate by means of identifiers such as public keys, signatures or shared passwords. As such, AKE allows two parties

Lorenzo Magliocco and Daniele Venturi were supported by project SERICS (PE00000014) and by project PARTHENON (B53D23013000006), under the MUR National Recovery and Resilience Plan funded by the European Union—NextGenerationEU. Daniele Venturi is member of the Gruppo Nazionale Calcolo Scientifico - Istituto Nazionale di Alta Matematica (GNCS-INdAM).

to establish a secure channel. Due to its sensitive nature, malicious actors may have a particular interest in undermining the security of AKE protocols (*e.g.*, by leaking the password of an honest party, or by establishing a shared key without authentication). To this extent, AKE protocols are typically designed in the setting of multi-party computation, where the adversary controls the communication channels and can corrupt some of the parties. Corrupted parties either simply follow the protocol (so-called *semi-honest* corruptions), or deviate arbitrarily from its intended execution (so-called *malicious* corruptions).

This threat model is widely adopted in the literature. However, it relies on the assumption of having access to uncorrupted parties that run the protocol exactly as prescribed. Unfortunately, as shown by the shocking Edward Snowden’s revelations, the latter assumption may not hold in practice, as the machine of an honest party could have been compromised in an undetectable manner, both in the case of its hardware (*e.g.*, via backdoored components) or its software (*e.g.*, via algorithm-substitution attacks, purposefully designed leaky constructions, or mistakenly instantiated protocols). Such undetectable corruptions enable an adversary to launch so-called subversion attacks, which may cause the target compromised machine to covertly exfiltrate information or behave in an unexpected manner upon receiving a specific triggering input.

A possible mitigation consists in equipping parties with *cryptographic reverse firewalls* (RFs), as first defined by Mironov and Stephens-Davidowitz [27]. These objects allow to sanitize inbound and outbound messages of the party they are attached to, thus destroying any potential side-channel while preserving functionality and security of the underlying protocol. The idea here is that protocol designers can instantiate parties and their respective RF on different physical machines on the same local network in order to achieve security in the presence of subversion attacks.

While the original formalism of [27] only accounted for standalone security, where each protocol is run in isolation, the setting of RFs has recently been extended to the universal composability (UC) framework by Chakraborty, Magri, Nielsen and Venturi [18]. The latter ensures that, once a designed protocol is proven to be secure, subversion resilience holds even if that protocol is arbitrarily composed with other protocols. This lifts the requirement of redoing the security analysis from scratch for each individual composition setting, thus yielding a modular design of subversion-resilient cryptographic protocols.

1.1 Password-Authenticated Key Exchange

In this work, we focus on instantiating Password-Authenticated Key Exchange (PAKE) in the subversion-resilient UC (srUC) framework [18], in which parties can derive a high-entropy secret key and verify their identities by means of a shared password. Given that passwords are considered to be low-entropy, the security definition of PAKE must take into account the fact that the adversary can guess the password with non-negligible probability. Thus, a protocol realizing PAKE is secure if no adversary is able to break it with probability better than guessing the password outright. Moreover, the PAKE functionality restricts

the ideal adversary to only perform *online* password guesses. In other words, the transcript of a PAKE protocol must not help the adversary to perform a dictionary (*i.e.*, offline) attack.

1.2 Our Results

Our main contribution consists in constructing the first UC PAKE protocol with security in the presence of subversion attacks, via RFs. Following [18], we consider a setting where each party is split into a core (which has secret inputs and is in charge of generating protocol messages) and a RF (which shares no secrets with the core and sanitizes the outgoing/incoming communication from/to the core using random coins). Both the core and the RF are subject to different flavours of corruption, modelling different kinds of subversion attacks.

In order to avoid simple impossibility results, we follow [18] and only consider the so-called *specious* subversions, in which a subverted core looks like an honest core to any efficient test, yet it may signal private information to the subverter via subliminal channels, or trigger an unexpected behaviour whenever a specific triggering message is received.

Our PAKE protocol is obtained by sanitizing the UC randomized equality protocol from oblivious transfer (OT) by Canetti *et al.* [12]. As an added bonus, this construction allows us to introduce several building blocks of independent interest in the srUC framework in a modular and natural manner. As we explain in the next section, essential changes to the original building blocks' design are needed, including the definition and the realization of sanitizable variants of intermediate ideal functionalities, new sanitation-friendly properties for cryptographic primitives, and extensions to the srUC model itself.

One difficulty in the realization of PAKE is that one *cannot* rely on authenticated channels. As shown by Barak *et al.* [7], this difficulty can be tackled generically by first designing a PAKE protocol assuming authenticated channels, and then compiling it into another protocol *without* authenticated channels using the concept of “split functionalities”. Such functionalities basically allow the adversary to disconnect parties completely, and engage in separate executions with each one of the two parties, where in each execution the adversary plays the role of the other party. We follow a similar recipe in the design of our PAKE protocol. In particular, we first realize subversion-resilient randomized equality, which is essentially PAKE with authenticated channels, assuming the existence of a functionality for sanitizable authenticated communication (which already appeared in [18], and is denoted by \mathcal{F}_{SAT}). Following [7], we then define a weaker split-authenticated (sanitizable) variant $s\mathcal{F}_{\text{SAT}}$ that allows the adversary to partition parties, and prove that a modification of their transformation allows to lift any protocol that multi-realizes a functionality \mathcal{F} assuming authenticated channels to one that realizes the corresponding “split version” (*i.e.*, $s\mathcal{F}$) without any assumption on channels, even in the presence of subversion.

In the process, we realize $s\mathcal{F}_{\text{SAT}}$ by sanitizing the protocol of [7, Section 4.2], introducing a new notion of *key-sanitizable* signature schemes with a matching security property. This improves on an open problem from [18], where the authors

were only able to realize \mathcal{F}_{SAT} assuming the presence of a PKI and by moving to a “three-tier model” variant of the framework, in which each party has an additional *operative* component that may only be honest or malicious. Even if used exclusively throughout the setup phase of the protocol, providing access to an operative component that is immune to subversion is a strong assumption that definitely weakens any result achieved in the framework: indeed, the three-tier model provides a trivial solution to counteract specious corruptions of the core for *any* functionality, as the operative is in principle allowed to run any protocol on behalf of the core. On the contrary, we realize the backbone of communication among components in the two-tier model without assuming a PKI, although only for the unauthenticated setting (*i.e.*, $s\mathcal{F}_{\text{SAT}}$).

Finally, we apply the aforementioned transformation to our randomized equality protocol, and realize subversion-resilient PAKE by constructing a protocol with access to the split version of the randomized equality functionality.

1.3 Technical Overview

Below, we provide an overview of the technical contributions, explaining the main ideas and tools behind our subversion-resilient PAKE protocol.

Sanitizing OT. Defining oblivious transfer in the presence of subversion attacks is a tricky task, as the (non-sanitized) functionality would allow a (specious) receiver to obtain exactly one of the inputs of the sender, which may act as a trigger if sampled maliciously. Similarly, it would allow a (specious) sender to sample the inputs in a leaky manner and send them over to a corrupted party. For this reason, in our sanitizable OT ideal functionality \mathcal{F}_{sOT} (depicted in Fig. 1), both firewalls are allowed to blind the sender’s inputs by means of a blinding operation. This way, the sender’s firewall can sanitize the sender’s randomly chosen inputs, and the receiver’s firewall can sanitize the inbound inputs.

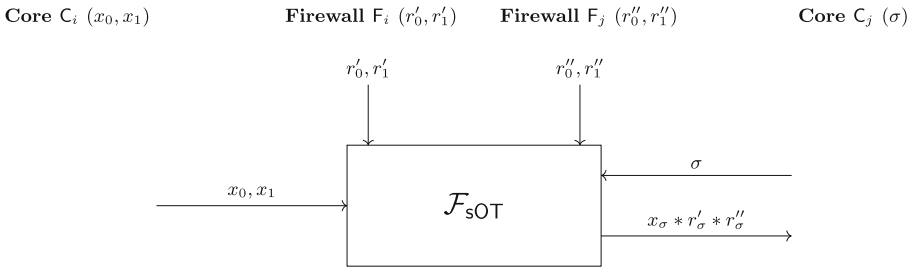


Fig. 1. Our sanitizable OT functionality \mathcal{F}_{sOT} , with $*$ being an appropriate blinding operation for the input domain.

Here, we introduce a different technique compared to that of the seminal framework. Namely, the functionality allows firewalls to explicitly contribute to the

sanitation, and disregard their contribution whenever the overall party related to that firewall is malicious. From a formal standpoint this is allowed, as there exists a corruption translation table that maps corruptions of individual components of a party to a corruption for the entire party, and currently the srUC framework only supports static corruptions, so the functionality knows in advance which parties are corrupted. This also makes sense for what concerns simulation: once we have mapped components to a malicious party we shouldn't simulate anything that occurs within that malicious party. As an example, while handling a malicious sender in a protocol realizing \mathcal{F}_{OT} , it suffices for the simulator to only forward to \mathcal{F}_{OT} the malicious sender's input messages. Indeed, the notion of blinding may not even be well-defined.

In order to instantiate \mathcal{F}_{OT} , we start by considering dual-mode cryptosystems as in Peikert *et al.* [28]. Briefly, in these cryptosystems the party holding the secret key specifies a decryption branch upon generating the keypair, and the party holding the public key specifies an encryption branch for each ciphertext. Decryption succeeds only for ciphertexts generated on the decryption branch. Moving to the subversion setting, we introduce a new primitive that we call *sanitizable homomorphic* dual-mode cryptosystems that extends dual-mode cryptosystems by additionally providing: (1) a procedure to carry out homomorphic operations on ciphertexts (*e.g.*, $\text{Enc}(m_1) * \text{Enc}(m_2) = \text{Enc}(m_1 * m_2)$), (2) a procedure to maul an encryption key pk to a different encryption key \tilde{pk} , and (3) a procedure to maul a ciphertext under encryption key \tilde{pk} to a ciphertext of the same message under encryption key pk . Looking ahead, item (1) allows firewalls to sanitize the messages input to the OT, and items (2, 3) allow to first blind a public key, introducing a layer of sanitation, and align encryptions accordingly, stripping that layer of sanitation away to preserve correctness. The construction from DDH of [28, Section 5] can be extended to verify our newly introduced properties in a straight-forward manner.

Finally, we instantiate the functionality by proposing an appropriate sanitation of the protocol of [28, Section 4], which unfolds as follows. The receiver produces a key pair that may only be used to decrypt values on the encryption branch matching the choice bit σ and sends the public key towards the sender. This key is sanitized once by each firewall. Upon receiving the (sanitized) key, the sender encrypts value x_b on encryption branch b , for $b \in \{0, 1\}$, and forwards these ciphertexts towards the receiver. Each firewall removes one layer of sanitation from the ciphertexts, so that the receiver can successfully decrypt the ciphertext on branch $b = \sigma$.

In the security proof, we first show that the construction is strongly sanitizing, *i.e.*, a specific core with a honest firewall is indistinguishable from an incorruptible core with a honest firewall, by using the aforementioned properties. After that, the simulation becomes extremely close to the one of the original protocol, as it leverages on the two (computationally indistinguishable) modes of the CRS to map the behaviour of the adversary to consistent queries to \mathcal{F}_{OT} .

We conclude the section by remarking that, exactly as in the original protocol of [28], it is possible to re-use the same CRS across multiple protocol runs. Hence,

we obtain a protocol that multi-realizes \mathcal{F}_{sOT} (*i.e.*, a protocol that realizes the the multi-session sanitizable OT functionality $\hat{\mathcal{F}}_{\text{sOT}}$).

Sanitizing Randomized Equality. Canetti *et al.* [12] instantiate the randomized equality functionality by proposing a protocol that relies on OT and roughly unfolds as follows: for an n -bit password, each party runs \mathcal{F}_{OT} n -times as the sender, inputting two random strings for each OT run, and n -times as the receiver, inputting the i -th bit of their password in the i -th run. Intuitively, the sender of each batch of OTs is able to choose the same random strings that were selected by the receiver only if the passwords are the same, and all these strings can be combined to derive a common shared key.

After defining \mathcal{F}_{sOT} , designing a protocol that realizes the randomized equality ideal functionality \mathcal{F}_{RE} in the subversion setting becomes immediate. In order to thwart information leakage originating from a biased sampling of the random strings, as well as inbound input-triggering strings, both firewalls blind the sender’s inputs in both OT batches with locally-sampled random strings. The trick to preserve correctness leverages on the symmetrical structure of the protocol: namely, random strings used for the i -th OT in which a core acts as the sender are re-used for the i -th OT in which the same core acts as the receiver.

Split Functionalities in the srUC Model. A PAKE protocol establishes (over an *unauthenticated* channel) a secret key among parties that share a common password. Thus, it makes little sense to build a PAKE protocol in a setting that already assumes the existence of authenticated channels.

The problem of achieving any form of secure computation (including protocols such as PAKE) in the UC unauthenticated channel setting was first described by Barak *et al.* [7]. In their setting, all the messages sent by parties can be tampered with and manipulated by the adversary unbeknownst to honest parties. The authors show that, while in this model it is not possible to achieve the same guarantees as with authenticated channels, meaningful security can still be provided: namely, the worst the adversary can do is split honest parties into independent execution sets before the protocol run, and act on behalf of all (honest) parties that are not within the same set. This way, even though honest parties can run the entire protocol with the adversary without even noticing it, they can rest assured that they will complete the entire run of the protocol interacting with the same set of parties since the start. In [7], this notion is captured in what the authors call *split functionalities*. One central result of [7] consists of showing a generic transformation for which any protocol UC n -realizing some n -party functionality \mathcal{F} relying on authenticated channels can be compiled into a similar protocol that UC-realizes the split functionality $s\mathcal{F}$, but now just relying on *unauthenticated* channels.

Given that [18] exclusively refers to authenticated channels, which are formalized with the “sanitizable authenticated transmission” functionality \mathcal{F}_{SAT} , in this work we extend the notion of split functionalities to the srUC model. More specifically, we show that the generic transformation of [7] for split protocols

carries over to our setting whenever the underlying *unauthenticated* channel is sanitizable. The latter notion is captured by the split version of \mathcal{F}_{SAT} , that we call $s\mathcal{F}_{\text{SAT}}$. This functionality allows the adversary to split parties in different authentication sets in a “link initialization” phase, before any message is exchanged. After that, the behaviour is exactly the same as \mathcal{F}_{SAT} , except that the adversary may deliver arbitrary messages to parties within different authentication sets.

A crucial component of the transformation is the construction of a protocol realizing $s\mathcal{F}_{\text{SAT}}$. For that, we introduce a new primitive that we call *key-sanitizable* signatures that: (1) provides a procedure to maul a verification key vk into $\tilde{\text{vk}}$, (2) a procedure to maul a signature under verification key vk into a signature under verification key $\tilde{\text{vk}}$, and (3) is equipped with a function f such that $f(\text{vk}_i, \tilde{\text{vk}}_j) = f(\tilde{\text{vk}}_i, \text{vk}_j)$, with $\tilde{\text{vk}}_i$ and $\tilde{\text{vk}}_j$ being verification keys mauled under the same randomness. We show that the BLS signature scheme [10] is a key-sanitizable signature scheme, with f being a bilinear map. In our protocol for $s\mathcal{F}_{\text{SAT}}$, parties exchange locally-generated keys, which are used to “initialize the link” by determining a session ID sid , and to sign messages that are exchanged through the link. Firewalls sanitize these keys and re-align signatures accordingly to preserve correctness, and the bilinear map allows parties to recompute the same sid in the presence of firewalls mauled the keys. We note however that the bilinear map restricts the protocol to the 2-party setting, which in turn restricts the transformation to only capture 2-party functionalities in the srUC model.

Once a protocol for $s\mathcal{F}_{\text{SAT}}$ is in place, one can simply white-box inspect the proofs of [7] and adapt them to the srUC setting. The core result is a lemma stating that any protocol 2-realizing a 2-party functionality \mathcal{F} in the wsrUC model assuming \mathcal{F}_{SAT} can be compiled into a protocol realizing $s\mathcal{F}$ in the wsrUC model assuming $s\mathcal{F}_{\text{SAT}}$. Given that any n -party functionality \mathcal{F} can be n -realized in the wsrUC model by the subversion-resilient GMW compiler of [18], we also obtain a theorem stating that any 2-party split functionality can be realized in the wsrUC model using only *unauthenticated* channels (in the $s\mathcal{F}_{\text{SAT}}$ -hybrid model), matching [7, Theorem 10]. As in traditional UC, a protocol poly-realizing a functionality roughly means that polynomially-many instances of that protocol may re-use the same setup.

The Final PAKE Protocol. At last, we combine all our ingredients together to realize PAKE in the subversion setting. First, we apply the split transformation to the protocol realizing \mathcal{F}_{RE} in the authenticated setting, obtaining a protocol that realizes $s\mathcal{F}_{\text{RE}}$ in the unauthenticated setting. Then, with a similar argument to that of Dupont *et al.* [22], we argue that $s\mathcal{F}_{\text{RE}}$ is sufficient to instantiate $\mathcal{F}_{\text{PAKE}}$. This can be shown by exhibiting a trivial protocol in the $s\mathcal{F}_{\text{RE}}$ -hybrid model that exclusively interacts with $s\mathcal{F}_{\text{RE}}$, and by showing that the power of splitting parties in $s\mathcal{F}_{\text{RE}}$ can be mapped to the power of performing password queries in $\mathcal{F}_{\text{PAKE}}$.

We observe that, as a corollary of the generic result of the previous paragraph, one also gets a protocol realizing $s\mathcal{F}_{\text{RE}}$ by relying on the srUC GMW compiler from [18], although with worse efficiency than our concrete construction from

DDH. For that, we provide a hand-wavy comparison of the two constructions by considering communication and round complexity.

Importantly, in this work we consider a PAKE functionality that only provides implicit rather than explicit authentication. This means that, while parties can be assured by the functionality that any other party capable of deriving the same session key must possess the password, there is no direct assurance that the counterpart has successfully computed the session key upon completion of the protocol. This decision was made for two primary reasons: (1) it streamlines our results, as explicit mutual authentication typically requires incorporating additional “key confirmation” steps at the protocol’s conclusion, which would complicate our protocol with the need for further sanitation processes, and (2) in many practical scenarios, such as secure channels, explicit authentication is not a requirement. Moreover, in our setting, mutual explicit authentication is inherently provided by any higher-level protocol that utilizes our PAKE as a foundation. For instance, in applications involving secure messaging, the act of successfully exchanging messages serves as explicit confirmation that both parties share the same session key.

Moreover, as a technical remark stemming from the srUC model, the PAKE functionality we realize implicitly includes the wrapper of [18] that simply adds dummy firewall parties in order to prevent trivial distinguishing from the environment. This also holds for \mathcal{F}_{RE} , but causes no differences in the behaviour of both functionalities. For a cleaner presentation and following [18], we omit the wrapper when using hybrid functionalities.

1.4 Related Work

Next, we discuss related works on the topics of reverse firewalls, subversion-resilient cryptography in general, and PAKE.

Reverse Firewalls and Subversion. Reverse firewalls were introduced by Mironov and Stephens-Davidowitz [27], who showed how to construct reverse firewalls for oblivious transfer (OT) and two-party computation with semi-honest security. Follow up works showed how to construct reverse firewalls for many other cryptographic primitives and protocols including: secure message transmission and key agreement [19, 21], interactive proof systems [24], and maliciously secure MPC for both the case of static [16] and adaptive [17] corruptions. However, most of these constructions lack modularity, as the security of each firewall is proven in isolation and does not extend to larger protocols when combined with other firewalls. This was addressed by Chakraborty, Magri, Nielsen and Venturi [18] with the proposal of the Subversion-Resilient Composability framework (srUC). The srUC allowed for the first time to build and to analyse subversion-resilient protocols under composition. [18] shows how to sanitize the classical GMW compiler [25] for MPC under subversion. Towards that, it also introduces the concept of sanitizable commitment and sanitizable commit-and-prove.

More recently and concurrently to this work, an alternative framework for subversion-resilient UC was put forward by Arnold *et al.* [4]. Compared to [18],

this new framework captures reverse firewalls in the plain UC model, but characterizes subversion by exclusively allowing an adversary to tamper with the function generating the randomness of a protocol. This rules out simple subversion attacks which [18] (and our paper) accounts for, such as having a specious core change its input to part of its secret state upon receiving a specific triggering value.

Ringerud [29] explored the problem of achieving subversion-resilient AKE in a standalone fashion (*i.e.*, without reverse firewalls or watchdogs), providing intuition on why realizing this primitive appears to be hard in such an adversarial setting.

Additional work on subversion includes algorithm substitution attacks [6, 9, 20], parameter subversion [2, 3, 8, 23], Cliptography [5, 31, 32], subliminal channels [33, 34] to list a few. We refer to [18, 27] for further related works, such as *watchdogs* and *self-guarding*.

PAKE. The seminal work by Canetti *et al.* [13] formalizes PAKE as an ideal functionality, and proposes an efficient protocol securely realizing this functionality in the setting of malicious corruptions and under *universal composability* [11], *i.e.*, when protocols can be arbitrarily composed with other protocols. The description was later extended to *explicit* mutual authentication in [12, 26], in which parties are able to tell whether they effectively authenticated or not. Our work is the first to achieve subversion-resilient PAKE in the UC framework.

1.5 Organization

In Sect. 2, we give a concise introduction to the subversion-resilient UC framework of [18]. In Sect. 3, we define and instantiate sanitizable oblivious transfer. In Sect. 4, we instantiate a subversion-resilient protocol for the randomized equality ideal functionality. In Sect. 5, we define and instantiate the sanitizable split-authenticated functionality, and port the transformation of Barak *et al.* [7] that allows to remove authenticated channels from our reference framework. In Sect. 6, we combine the results of previous sections to achieve subversion-resilient PAKE. Finally, in Sect. 7, we conclude the paper with a few related open problems for further research. See Fig. 2 for a visual representation of how our results are linked to one another.

2 A Brief Recap of Subversion-Resilient UC

We give a brief overview of subversion resilience in the UC framework (srUC for short). We refer the reader to [18] for further details, and to [11] for a complete treatment of the UC framework.

Authenticated Communication (\mathcal{F}_{SAT}) Unauthenticated Communication ($s\mathcal{F}_{\text{SAT}}$)

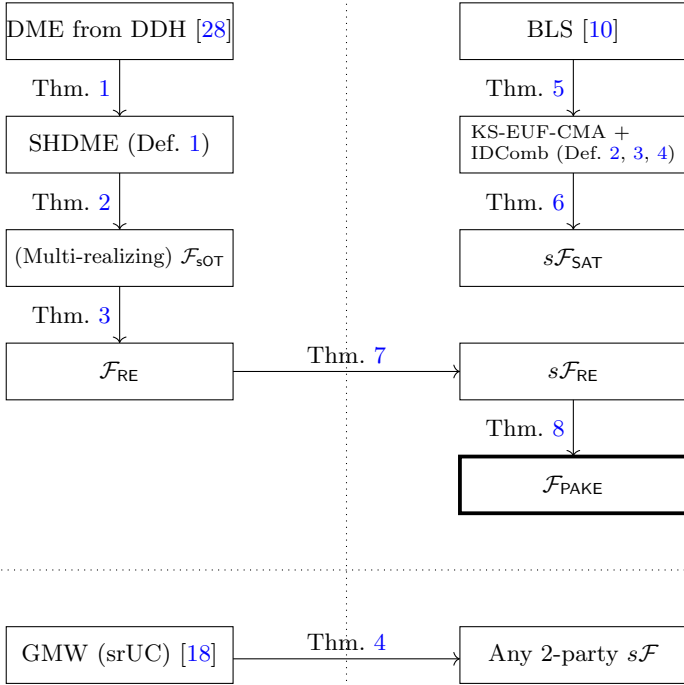


Fig. 2. A visual summary of the contributions of this paper. All the functionalities are realized in the srUC framework of [18]. DME stands for Dual-Mode Encryption. SHDME stands for Sanitizable Homomorphic DME. KS-EUF-CMA stands for Key-Sanitizable EUF-CMA. IDComb is a shorthand for Consistent Identity Combinability.

2.1 Corruption Types

Each party P_i in the protocol is modelled as two independent parties: a core C_i , which hosts the code associated with the protocol (and may contain secrets), and a firewall F_i , which may intervene on all the messages associated with their respective core (both inbound and outbound). Since cores and firewalls are independent parties, they may also be corrupted independently. The model of [18] specifies that the relevant corruption cases for the core are HONEST, MALICIOUS, or SPECIOUS, while the ones for the firewall are HONEST, SEMIHONEST, or MALICIOUS. Mapping the corruption possibilities for the parties $P_i = (C_i, F_i)$ in a regular UC functionality gives rise to the following corruption translation table (Table 1):

Table 1. The corruption translation table of [18].

Core C	Firewall F	Party P in \mathcal{F}
HONEST	SEMIHONEST	HONEST
SPECIOUS	HONEST	HONEST
HONEST	MALICIOUS	ISOLATE
MALICIOUS	MALICIOUS	MALICIOUS

Specious Corruption. A specious corruption is a type of subversion where the subverted core looks indistinguishable from the honest core to any efficient test. The main idea is that we consider corruptions where a core C_i has been replaced by another implementation \tilde{C}_i which cannot be distinguished from C_i by black-box access to \tilde{C}_i or C_i . Intuitively, a specious corruption can be thought of as a subversion that remains undetectable.

Isolate Corruption. ISOLATE is a weaker type of corruption that models the setting where a malicious firewall simply cuts the communication of an honest core with the outside world. This is typically modelled as a MALICIOUS corruption in the authenticated setting, and as a MITM attack in the unauthenticated setting, and can therefore be safely dropped from the analysis.

Strong Sanitization. A firewall is strongly sanitizing if an adversary is unable to distinguish an execution of the protocol with a specious core equipped with an honest firewall from an execution of the protocol with an honest core equipped with an honest firewall. As shown in [18], whenever the firewalls are strongly sanitizing, the SPECIOUS core and HONEST firewall case is the same as considering an HONEST core and an HONEST firewall.

2.2 Ideal Functionalities

There are two types of ideal functionalities in srUC: *sanitizable* functionalities and *regular* functionalities. Sanitizable functionalities are the ones where cores and firewalls explicitly interact with the functionality. For that, sanitizable functionalities expose, for each party P_i , an input-output interface IO_i that interacts with the core C_i , and a sanitation interface S_i that interacts with the firewall F_i . Regular functionalities have the same flavor of the functionalities used in the UC framework, where the functionality will only communicate with parties and is not aware of cores and firewalls. The goal of considering regular functionalities is that it is perfectly valid and desirable to be able to build protocols that realize a regular functionality (*e.g.*, coin tossing) under subversion attacks. However, since there is no support for sanitation interfaces in regular functionalities, the model considers a wrapped version of the functionality \mathcal{F} , denoted by $\text{Wrap}(\mathcal{F})$, that handles all the boilerplate code of translating the combinations of corruptions of cores and firewalls to corruptions of parties in \mathcal{F} . The wrapper also passes any message coming from the functionality and directed to party P_i to the corresponding core C_i and firewall F_i , and it is needed to avoid trivial

distinguishing attacks in the UC framework, since the actual protocol will be implemented with cores and firewalls. For what concerns security definitions, two separate notions are presented in [18], according to the type of functionality that is being realized: subversion-resilient UC (srUC) security for *sanitizable* ideal functionalities, and *wrapped* subversion-resilient UC (wsrUC) security for *regular* ideal functionalities. We refer the reader to [18] for the formal definitions and further details.

2.3 Communication Channels

In all the protocols of [18], communication is mediated by a sanitizable ideal functionality for authenticated communication \mathcal{F}_{SAT} , which fundamentally embeds three capabilities:

- It allows to distribute a setup (e.g., a CRS) by means of a Setup algorithm.
- It provides *secure* channels between cores and their respective firewall.
- It provides *authenticated* channels between firewalls.

In what follows, we report a variant of the description of \mathcal{F}_{SAT} that does *not* include the first capability. This is a design choice that allows to better separate setup and communication: indeed, the former may be captured by a separate ideal functionality \mathcal{F}_{crs} .

Functionality \mathcal{F}_{SAT}

- On input (SEND, P_i, P_j, a) on IO_i , it forwards the tuple on S_i . As in the original description, we assume that a is sent at most once from honest parties.
- On input (SEND, P_i, P_k, b) on S_i , it leaks the tuple to the adversary \mathcal{S} , and internally stores the tuple.
- On input (DELIVER, (SEND, P_i, P_k, b)) from the adversary, where the SEND tuple is stored, it outputs (RECEIVE, P_i, P_k, b) on S_k and deletes the tuple.
- On input (RECEIVE, P_i, P_m, c) on S_m , it outputs (RECEIVE, P_i, P_m, c) on IO_m .

An important observation is that \mathcal{F}_{SAT} induces a core-to-core authenticated channel. While this is an acceptable backbone of communication for our protocols in Sects. 3 and 4, it makes little sense to instantiate PAKE by already assuming authenticated channels. In Sect. 5, we overcome this limitation by defining a weaker functionality $s\mathcal{F}_{\text{SAT}}$ that models the unauthenticated setting by allowing the adversary to partition parties, in line with the work of Barak *et al.* [7].

3 Sanitizing Oblivious Transfer

In this section, we first propose a *sanitizable* ideal functionality for oblivious transfer that will be used as a building block for the sanitation of randomized

equality in Sect. 4. Secondly, we recap dual-mode cryptosystems, define *sanitizable homomorphic* dual-mode cryptosystems, and exhibit an instantiation for this new primitive from the DDH assumption. We use the latter notion to sanitize the generic framework for OT of Peikert *et al.* [28], obtaining a protocol for the *sanitizable* oblivious transfer functionality \mathcal{F}_{OT} . Finally, we argue that, in line with the instantiation of [28], our protocol can reuse the same CRS across multiple runs, thus realizing the multi-session extension of \mathcal{F}_{OT} (also denoted by $\hat{\mathcal{F}}_{\text{OT}}$).

3.1 Sanitizable OT

Following the ideas presented in the technical overview in Sect. 1.3, we describe *sanitizable* ideal functionality for oblivious transfer \mathcal{F}_{OT} , in which both firewalls may intervene in the sanitation of the sender's inputs.

Functionality \mathcal{F}_{OT}

\mathcal{F}_{OT} is a sanitizable ideal functionality that interacts with the sender $S = (C_S, F_S)$ and the receiver $R = (C_R, F_R)$, parameterized by input domain $\mathcal{I} \subseteq \{0, 1\}^n$ and a blinding operation $*$: $\mathcal{I}^2 \rightarrow \mathcal{I}$.

Interface IO_i :

Upon receiving a query (SENDER, sid, (x_0, x_1)) from C_S on IO_S :

Record (SENDER, sid, (x_0, x_1)) and forward the tuple on S_i . Ignore subsequent commands of the form (SENDER, sid, \cdot).

Upon receiving a query (RECEIVER, sid, σ) from C_R on IO_R :

Check if a record (SENDER, sid, (\hat{x}_0, \hat{x}_1)) exists. If this is the case, check the following:

- * The message (BLIND, sid, \cdot) was sent to \mathcal{F}_{OT} on S_S . If S is malicious according to the corruption translation table, mark this check as passed.
- * The message (BLIND, sid, \cdot) was sent to \mathcal{F}_{OT} on S_R . If R is malicious according to the corruption translation table, mark this check as passed.

If the conditions above hold, output (sid, \hat{x}_σ) to R , sid to the adversary S , and halt. Otherwise, send nothing to R but continue running.

Interface S_i :

Upon receiving a query (BLIND, sid, (x'_0, x'_1)) from F_S on S_S :

If S is malicious according to the corruption translation table, do nothing. Otherwise, check if a record (SENDER, sid, (x_0, x_1)) exists. If so, update the tuple to (SENDER, sid, $(\tilde{x}_0, \tilde{x}_1)$), with $\tilde{x}_b = x_b * x'_b$. Otherwise, do nothing. Ignore future commands of the form (BLIND, sid, \cdot) on S_S .

Upon receiving a query (BLIND, sid, (x''_0, x''_1)) from F_R on S_R :

If R is malicious according to the corruption translation table, do nothing. Otherwise, check the following:

- * A record (SENDER, sid, $(\tilde{x}_0, \tilde{x}_1)$) exists.

* A message $(\text{BLIND}, \text{sid}, \cdot)$ was sent to \mathcal{F}_{SOT} on \mathbb{S}_S . If S is malicious according to the corruption translation table, mark this check as passed.

If the conditions above hold, update the tuple to $(\text{SENDER}, \text{sid}, (\hat{x}_0, \hat{x}_1))$, with $\hat{x}_b = \hat{x}_b * x'_b$. Otherwise, do nothing. Ignore future commands of the form $(\text{BLIND}, \text{sid}, \cdot)$ on \mathbb{S}_R .

The ideal functionality is parameterized by a blinding operation $*$, which may be tailored to the input domain of choice (*e.g.*, for additive blinding, $x_0 * x'_0 = x_0 \oplus x'_0$; for multiplicative blinding, $x_0 * x'_0 = x_0 x'_0$). Furthermore, the functionality disregards blinding inputs from firewalls of parties that, according to the corruption translation table, are malicious. As discussed throughout the technical overview in Sect. 1.3, this is reasonable: the corruption status of individual components can be determined in advance (as we are in the static setting), and their combined behaviour can be considered as a single party by following the corruption translation table. If the joint party is malicious, we do not have to simulate anything related to messages internally exchanged by the adversary. In particular, the blinding operation may not be well-defined at all.

3.2 Sanitizable Homomorphic Dual-Mode Encryption

Dual-mode cryptosystems operate like traditional public-key cryptosystems, except for the following differences. First, they introduce the notion of *encryption branches*, for which the key generation algorithm takes as an additional input a branch $\sigma \in \{0, 1\}$. The party holding the public key can choose either branch $b \in \{0, 1\}$ over which to encrypt a message. The party holding the secret key is able to decrypt the ciphertext successfully only if $\sigma = b$. Second, they rely on a common-reference string that may be setup either in *messy* mode or *decryption* mode. These modes are computationally indistinguishable and induce different algorithms for the generation of a trapdoor, yielding different security guarantees: in messy mode, the sender has statistical security and the receiver has computational security, whereas in decryption mode the security properties are mirrored. We refer the reader to [28, Section 3] for the formal definition and further details.

Sanitizable Homomorphic Dual-Mode Cryptosystems. Looking ahead, we need to augment dual-mode cryptosystems to allow the sanitation of public keys, ciphertexts, and plaintexts related to ciphertexts. For that, we formally define *sanitizable homomorphic* dual-mode cryptosystems in what follows.

Definition 1 (Sanitizable Homomorphic Dual-Mode Cryptosystems).

A *sanitizable homomorphic dual-mode cryptosystem* consists of a tuple of algorithms $(\text{Setup}, \text{KeyGen}, \text{Enc}, \text{Dec}, \text{FindMessy}, \text{TrapKeyGen}, \text{HomOp}, \text{MaulPK}, \text{AlignEnc})$ with the following properties:

1. **Dual-mode cryptosystem:** The tuple of algorithms (Setup, KeyGen, Enc, Dec, FindMessy, TrapKeyGen) constitutes a dual-mode cryptosystem.
2. **Homomorphic ciphertexts:** For every $\sigma \in \{0, 1\}$, for every $(pk, sk) \leftarrow_{\S} \text{KeyGen}(\sigma)$, for every $c_i \leftarrow_{\S} \text{Enc}(pk, \sigma, m_i)$, with $i \in \{0, 1\}$ and $m_i \in \{0, 1\}^n$, $\text{HomOp}(m_0, m_1)$ produces a new ciphertext of message $m_0 * m_1$, i.e., $\text{HomOp}(c_0, c_1) = \text{Enc}(pk, \sigma, (m_0 * m_1))$.
3. **Consistent key sanitation:** For every $\sigma \in \{0, 1\}$, for every $(pk, sk) \leftarrow_{\S} \text{KeyGen}(\sigma)$, for every $\rho \in \{0, 1\}^n$, $\text{MaulPK}(pk, \rho)$ outputs a new encryption key \widetilde{pk} with the following property. For every $\tilde{c} \leftarrow_{\S} \text{Enc}(\widetilde{pk}, \sigma, m)$, with $i \in \{0, 1\}$ and $m \in \{0, 1\}^n$, $\text{AlignEnc}(c, \rho)$ produces a new ciphertext c under public key pk , i.e., $\text{AlignEnc}(\tilde{c}, \rho) = c$, where $c = \text{Enc}(pk, \sigma, m)$.

Intuitively, MaulPK and AlignEnc are defined as a (symmetric) tuple of algorithms as firewalls will first sanitize the outbound encryption key by running MaulPK with some randomness. Then, upon receiving any ciphertext encrypted under the new mauled public key, the firewall will “strip” the layer of sanitation by using the same randomness used for MaulPK, outputting a ciphertext containing the same message for the non-mauled public key pk .

Remark 1. The MaulPK, AlignEnc, HomOp algorithms are outputting keys and ciphertexts implicitly combining the randomness of their inputs. This is essential in the context of sanitation, as it allows a firewall to run these algorithms to combine their “good randomness” to destroy subliminal channels stemming from values with “bad randomness” output by their core.

Instantiation from DDH. We briefly recap the instantiation of dual-mode cryptosystems from DDH of [28, Section 5]. In what follows, we denote \mathbb{G} as the group description on a cyclic group G of prime order p for which DDH is hard, with generators g, h .

- The CRS is a tuple (g_0, h_0, g_1, h_1) , with different trapdoors according to the mode of operation.
- $\text{KeyGen}(\sigma) = ((g_{\sigma}^r, h_{\sigma}^r), r) = ((pk_1, pk_2), sk) = (pk, sk)$.
- $\text{Enc}(pk, m, b) = (g_b^s h_b^t, pk_1^s pk_2^t m) = (c_1, c_2)$.
- $\text{Dec}(sk, c) = c_2 / c_1^r$.

The DDH cryptosystem is compatible with all the additional interfaces we introduced in Definition 1, and we define algorithms matching the newly introduced properties in a straight-forward manner:

- $\text{MaulPK}(pk, \rho)$: Output pk^{ρ} .
- $\text{AlignEnc}(c, \rho)$: Parse $c = (c_1, c_2)$. Output $\tilde{c} = (c_1^{\rho}, c_2)$.
- $\text{HomOp}(c_0, c_1)$: Output $c_0 c_1$.

Theorem 1. *The DDH cryptosystem of [28] with the additional algorithms specified above is a sanitizable homomorphic dual-mode cryptosystem, assuming that DDH is hard for \mathbb{G} .*

The theorem follows by inspection of the newly-introduced algorithms. A formal proof is given in the full version.

3.3 A Generic Framework for Sanitizable OT

As shown in the generic framework of [28, Section 4], having access to a dual-mode cryptosystem allows the instantiation of \mathcal{F}_{OT} in a natural manner: the receiver uses its choice bit σ as the selected decryption branch, and the sender encrypts each of its inputs x_b on a separate encryption branch $b \in \{0, 1\}$. The receiver will only be able to decrypt the ciphertext on branch $\sigma = b$.

Sanitizing the Framework. From a high-level perspective, our sanitized protocol leverages homomorphic ciphertexts to blind the sender’s inputs and uses consistent key sanitation to sanitize the receiver’s outbound encryption key and realign the inbound ciphertexts for decryption purposes. These operations also destroy any potential subliminal channel linked to the original ciphertexts or to the keys. In Fig. 3, we depict a protocol run showing only the firewall of the sender, since the firewall of the receiver behaves exactly in the same way.

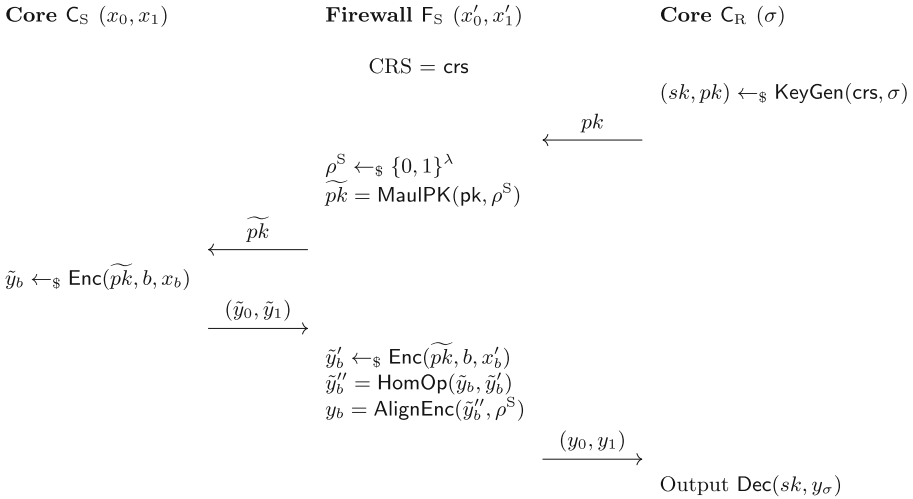


Fig. 3. A sanitation of the generic framework of Peikert *et al.* [28], realizing \mathcal{F}_{sOT} . The receiver’s firewall is omitted, as it runs the same code as F_S .

Theorem 2. *The protocol in Fig. 3, parameterized by mode $\in \{\text{mes}, \text{dec}\}$, realizes the sanitizable functionality \mathcal{F}_{sOT} in the $(\mathcal{F}_{\text{SAT}}, \mathcal{F}_{\text{crs}})$ -hybrid model under static corruptions. For mode = mes, the sender’s security is statistical and the receiver’s security is computational; for mode = dec, the security properties are reversed.*

Intuitively, we first show that the firewalls are able to thwart all subversion attacks (both inbound and outbound). Then, we simulate similarly to the original proof, with the twist that we do not have to simulate inputs of malicious parties (as per the considerations in the technical overview). We defer the formal proof to the full version.

3.4 Multi-session \mathcal{F}_{SOT}

Informally, a multi-session ideal functionality in UC is an ideal functionality that allows “multiple runs” of the functionality using the *same setup*. As a concrete example, the commitment functionality \mathcal{F}_{COM} allows a committer to commit to a single value; to produce another commitment a new and independent instance of (the protocol realizing) \mathcal{F}_{COM} must be spawned with a brand new setup. In contrast, the multi-session functionality $\mathcal{F}_{\text{MCOM}}$ allows a committer to perform poly-many commitments using the same setup. Hence, using multiple instances of $\mathcal{F}_{\text{MCOM}}$ has the same effect as using a single instance of $\mathcal{F}_{\text{MCOM}}$.

Moving to our case, we note that the generic framework of [28] actually realizes the multi-session version of \mathcal{F}_{OT} (also denoted as $\hat{\mathcal{F}}_{\text{OT}}$). Given that our protocol in Fig. 3 has the same structure as the protocol of [28], we observe that we can reuse the same CRS across multiple runs, each with a distinct sub-session ID. The presence of subverted cores does not impact this property, as the sanitation operated from the firewalls uses independently-sampled random strings for each sub-protocol run.

4 Sanitizing Randomized Equality

In this section, we present our sanitized protocol for the (regular) randomized equality ideal functionality \mathcal{F}_{RE} that relies on authenticated channels (i.e., \mathcal{F}_{SAT}) and \mathcal{F}_{SOT} , following the construction of Canetti *et al.* [12].

4.1 Description of \mathcal{F}_{RE}

We describe a variation of the randomized equality ideal functionality \mathcal{F}_{RE} of [12], with technical improvements from Dupont *et al.* [22].

Functionality \mathcal{F}_{RE}

The functionality \mathcal{F}_{RE} is parameterized by a security parameter λ . It interacts with an initiator $I = (C_I, F_I)$, a responder $R = (C_R, F_R)$, and the adversary \mathcal{S} via the following messages:

Upon receiving a query (NEWSESSION, sid , I , R , w^I), **from I:**

Record (I, R, w^I) and send a message (sid, I, R) to \mathcal{S} . Ignore all future messages from I .

Upon receiving a query (OK, sid) **from S:**

Send a message (WAKEUP, sid , I , R) to R . Ignore all future (OK) messages.

Upon receiving a query (RESPOND, sid , I , R , w^R) **from R:**

- If $w^R = w^I$, choose $\text{skey} \leftarrow_{\mathcal{S}} \{0, 1\}^\lambda$ and store $\text{skey}_I = \text{skey}_R = \text{skey}$.
- If $w^R \neq w^I$, then set $\text{skey}_I \leftarrow_{\mathcal{S}} \{0, 1\}^\lambda$, $\text{skey}_R \leftarrow_{\mathcal{S}} \{0, 1\}^\lambda$.

In both cases, ignore subsequent inputs from R .

Upon receiving a query (NEWKEY, sid , P , K), $P \in \{I, R\}$ **from S:**

- If any of the following conditions hold, output (sid, K) to party P :
 - P is corrupted.

- $w^I = w^R$, and the peer of P is corrupted.
 - Otherwise, output $(\text{sid}, \text{skey}_P)$ to party P.
- Ignore all subsequent (NEWKEY, P) queries for the same party P.

4.2 Randomized Equality from OT

We sanitize the RE from OT protocol of [12, Section 2.2] by using \mathcal{F}_{sOT} , restricting to implicit mutual authentication as per the considerations in the technical overview. Compared to the non-sanitized protocol, we parameterize the input domain \mathcal{I} and the respective blinding operation $*$, in line with the description of \mathcal{F}_{sOT} . For ease of exposition, we depict the protocol in Fig. 4 assuming 1-bit passwords. The n -bit password case runs exactly in the same way except that (i) it uses n OTs within the multi-session sanitizable OT functionality $\hat{\mathcal{F}}_{\text{sOT}}$, and (ii) it computes keys using operator $*$ with n random strings rather than only one. In order to preserve correctness, we leverage the symmetry of the protocol. In particular, the values each party retrieves from the batch of OTs in which they act as receivers embeds the random strings that are used by both firewalls, and these strings are the same also for the other OT batch. This also thwarts both input triggering attacks, as well as information leakage.

Theorem 3. *The protocol in Fig. 4 $w\text{s}r\text{UC}$ -realizes the \mathcal{F}_{RE} ideal functionality in the $(\mathcal{F}_{\text{sOT}}, \mathcal{F}_{\text{sAT}})$ -hybrid model under static corruptions.*

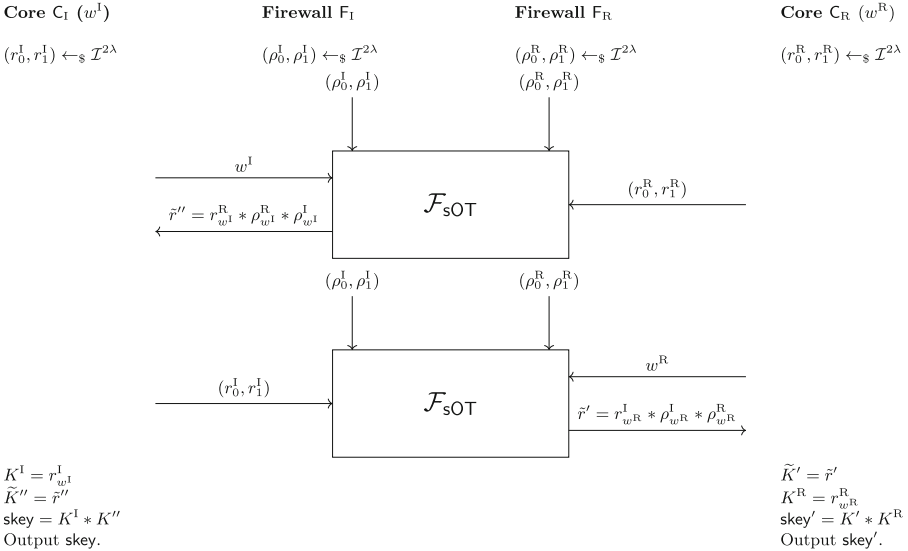


Fig. 4. A sanitizing protocol for \mathcal{F}_{RE} from sanitizable OT with a 1-bit password.

Within the proof, we first show strong sanitation of firewalls, and then proceed similarly to [12]. We defer the formal proof and an explicit analysis of correctness to the full version.

5 Subversion-Resilient Split Functionalities

In this section, we extend the notion of split functionalities of Barak *et al.* [7] to the srUC framework. Informally, we want to show that, for *any* well-formed¹ regular 2-party² ideal functionality \mathcal{F} , there exists a protocol that realizes the 2-party $s\mathcal{F}$ functionality with wsrUC-security in the CRS model. More formally, the goal of this section consists in proving an adaptation of [7, Theorem 10] to our setting, *i.e.*:

Theorem 4. *Let \mathcal{F} be a (regular) 2-party UC functionality. Then, assuming key-sanitizable signatures with consistent identity combinability, there exists a protocol that securely realizes the 2-party split functionality $s\mathcal{F}$ in the wsrUC model.*

Towards that, we follow the same strategy as [7] and proceed in the following three stages:

- *Link initialization:* The first step consists in building the *sanitizable* split-authenticated functionality $s\mathcal{F}_{\text{SAT}}$ that parties will use to communicate. The $s\mathcal{F}_{\text{SAT}}$ functionality can be seen as the split version of the \mathcal{F}_{SAT} functionality.
- *Multi-session security:* As the second step, we show that when authenticated channels are available, any functionality can be “poly-realized” in the wsrUC model. Here, poly-realizing a functionality informally means that security of the protocol implementing the functionality still holds even when multiple (*i.e.*, poly-many) instances of the protocol share the *same setup*. For that, we show that the subversion-resilient GMW protocol from [18] poly-realizes any functionality in the wsrUC model.
- *Unauthenticated channels:* Finally, we adapt the generic transformation of [7] that transforms any protocol π that 2-realizes a 2-party functionality \mathcal{F} given authenticated channels (*i.e.*, \mathcal{F}_{SAT}) in the wsrUC model into a protocol that realizes $s\mathcal{F}$ given access to $s\mathcal{F}_{\text{SAT}}$ in the wsrUC model.

Next, we look at each of these stages individually towards demonstrating Theorem 4.

¹ The “well-formed” property is to rule out unrealistic functionalities as explained in [7, 15].

² We restrict our attention to 2-party functionalities (in contrast to [7]) as the theorem relies on the sanitizable $s\mathcal{F}_{\text{SAT}}$ functionality that we only show how to realize for the 2-party setting.

5.1 Building Link Initialization

In this section we formally define $s\mathcal{F}_{\text{SAT}}$ (*i.e.*, the split version of the sanitizable authenticated channel functionality \mathcal{F}_{SAT} of [18]) and build a protocol that realizes it in the 2-party setting in the srUC model. For that, we introduce the notion of *key-sanitizable* signatures and show that it can be instantiated with the BLS signature scheme [10].

Description of $s\mathcal{F}_{\text{SAT}}$. The $s\mathcal{F}_{\text{SAT}}$ functionality has a similar structure to \mathcal{F}_{SAT} , with the addition of having a link initialization phase. In contrast with \mathcal{F}_{SAT} , the only guarantee provided by the functionality is that each party will be interacting with the same entity throughout the entire protocol run, but that entity could either be the expected party or the adversary itself. We describe $s\mathcal{F}_{\text{SAT}}$ next.

Functionality $s\mathcal{F}_{\text{SAT}}$

$s\mathcal{F}_{\text{SAT}}$ is a sanitizable ideal functionality that interacts with an adversary \mathcal{S} and a set of parties, each composed of a core C and a firewall F . The functionality consists of the following communication interfaces.

Initialization

- Upon activation with input $(\text{INIT}, \text{sid})$ from party P : Parse $\text{sid} = (\mathcal{P}, \text{sid}')$ where \mathcal{P} is a set of parties that includes P . Forward $(\text{INIT}, \text{sid}, P)$ to the adversary \mathcal{S} .
- Upon receiving the message $(\text{INIT}, \text{sid}, P, H, \text{sid}_H)$, from \mathcal{S} : Verify that $H \subseteq \mathcal{P}$, that the list H of party identities includes $P = (C, F)$, and that for all recorded sets H' either (i) $H \cap H'$ contains only corrupted parties (as per the standard corruption transition table in Table 1) and $\text{sid}_H \neq \text{sid}_{H'}$, or (ii) $H' = H$ and $\text{sid}_H = \text{sid}_{H'}$. If any of the check fails, do nothing. Otherwise, output $(\text{INIT}, \text{sid}, \text{sid}_H)$ to P and record (H, sid_H) if not yet recorded.

Message Authentication

- Upon receiving the message $(\text{SEND}, \text{sid}, P_i, P_j, m)$ on IO_i where $P_j \in \mathcal{P}$: Output the tuple on S_i .
- Upon receiving the message $(\text{SEND}, \text{sid}, P_i, P_j, \tilde{m})$ on S_i : Add the tuple to an (initially empty) list \mathcal{W} of waiting messages. The same tuple can appear multiple times in the list. Then, leak the tuple to \mathcal{S} .
- Upon receiving the message $(\text{DELIVER}, (\text{SEND}, \text{sid}, P_i, P_j, \tilde{m}))$ from \mathcal{S} :
 - If P_j did not previously receive an $(\text{INIT}, \text{sid}, \text{sid}_H)$ output, do nothing.
 - Else, if P_i is in the authentication set H of P_j , and P_i is uncorrupted, then: if there is a tuple $(\text{SEND}, \text{sid}, P_i, P_j, \tilde{m}) \in \mathcal{W}$, remove one appearance of the tuple from \mathcal{W} and output $(\text{RECEIVE}, \text{sid}, P_i, P_j, \tilde{m})$ on S_j . Otherwise, do nothing.
 - Else (*i.e.*, P_j received $(\text{INIT}, \text{sid}, \text{sid}_H)$, and either P_i is corrupted or $P_i \notin H$), output $(\text{RECEIVE}, \text{sid}, P_i, P_j, \tilde{m})$ on S_j , regardless of \mathcal{W} .
- Upon receiving the message $(\text{RECEIVE}, \text{sid}, P_i, P_j, \tilde{m})$ on S_j , output the tuple on IO_j .

The functionality consists of a preliminary initialization phase and the actual message authentication phase. In the initialization phase, the adversary controls how parties will be partitioned in the respective authentication sets. Intuitively, parties within the same authentication set will be able to communicate as if there was an authenticated channel between them. It is however possible for the adversary to participate in different authentication sets on behalf of all corrupted parties and any party outside of that authentication set. In the message authentication phase, honest parties will transmit messages in an authenticated fashion within the same authentication set. However, they may very well receive messages out of the blue from the adversary on behalf of any party that is corrupted or outside the authentication set.

With respect to sanitation, whenever a core sends a message m with destination P_j on IO_i , the message is output on S_i . This means that m is output to a firewall that will decide if/how to sanitize m to \tilde{m} in any arbitrary way, without involving the functionality in the sanitation process. Once the firewall determines the message \tilde{m} to send to P_j , \tilde{m} is leaked to the adversary. According to the partition of parties performed in the link authentication phase, the adversary has different capabilities:

- If the recipient party is within the same authentication set, the message is added to a message queue, and the adversary can exclusively control its delivery time. This behaviour is indeed equivalent to \mathcal{F}_{SAT} , in which the message is stored and then output to the recipient party whenever the adversary decides to do so.
- If P_i is corrupted or the parties are in different authentication sets, the adversary may deliver arbitrary messages to P_j , disregarding the message queue.

Whenever the adversary allows the delivery of a message, that message is output to the firewall F_j . Similarly to the sending phase, F_j may now modify the message arbitrarily without involving the functionality. Once a (potentially different) message \hat{m} is determined by F_j , it is delivered by the functionality to C_j .

We stress that, as it is the case for \mathcal{F}_{SAT} , cores and their respective firewall are allowed to freely communicate through secure channels. This is achieved by means of SEND messages (from a core to its firewall), and RECEIVE messages (from a firewall to its core). In principle, a firewall may send back any message to its core, even if it was not related to any DELIVER message from the adversary.

Key-Sanitizable Signature Schemes. In the construction of \mathcal{F}_{SA} of [7, Section 4.2], parties exchange locally-generated keys and sign their messages in order to preserve the split-authenticated security of the communication channel. However, in order to avoid subversion attacks, both inbound and outbound verification keys have to be appropriately sanitized by firewalls, breaking correctness in the verification of the signature. In order to overcome this limitation, we introduce a new notion that we call *key-sanitizable* signature schemes.

Informally, a *key-sanitizable* signature scheme allows to maul the verification key from vk to $\tilde{\text{vk}}$ by means of an algorithm MaulVK that takes as input

randomness ρ . The same randomness may be re-used by an algorithm AlignSig to align an (accepting) signature σ produced under secret key sk , producing a signature $\tilde{\sigma}$ that verifies with mauled key $\tilde{\text{vk}}$. The latter operation should also be invertible, meaning that the signature σ may be re-computed from $\tilde{\sigma}$ and ρ . We formally define this notion as a natural extension of traditional signatures in Definition 2, introducing a matching security notion in Definition 3 that extends EUF-CMA security to account for the newly introduced algorithms. This new security notion is implied in a black-box manner by any EUF-CMA scheme supporting the aforementioned algorithms.

Definition 2 (Key-sanitizable signature scheme). *A key-sanitizable signature scheme consists of a tuple of polynomial-time algorithms $(\text{KeyGen}, \text{Sign}, \text{Vrfy}, \text{MaulVK}, \text{AlignSig}, \text{UnAlignSig})$ with the following properties:*

1. **Correctness:** For every $(\text{vk}, \text{sk}) \leftarrow_{\S} \text{KeyGen}(1^\lambda)$, for every $\sigma \leftarrow_{\S} \text{Sign}(\text{sk}, m)$ with $m \in \{0, 1\}^n$, $\text{Vrfy}(\text{vk}, (m, \sigma)) = 1$.
2. **Consistent key sanitation:** For every $(\text{vk}, \text{sk}) \leftarrow_{\S} \text{KeyGen}(1^\lambda)$, for every $\rho \in \{0, 1\}^n$, $\text{MaulVK}(\text{vk}, \rho)$ outputs a new verification key $\tilde{\text{vk}}$ with the following property. For every $\sigma \leftarrow_{\S} \text{Sign}(\text{sk}, m)$ with $m \in \{0, 1\}^n$, $\text{AlignSig}((\text{vk}, \sigma, m), \rho)$ produces an accepting signature $\tilde{\sigma}$ for message m verifiable by verification key $\tilde{\text{vk}}$, i.e., $\text{Vrfy}(\tilde{\text{vk}}, \text{AlignSig}((\text{vk}, \sigma, m), \rho)) = 1$, where $\tilde{\text{vk}} = \text{MaulVK}(\text{vk}, \rho)$ and $\sigma = \text{Sign}(\text{sk}, m)$.
3. **Alignment invertibility:** For every $(\text{vk}, \text{sk}) \leftarrow_{\S} \text{KeyGen}(1^\lambda)$, for every $\sigma \leftarrow_{\S} \text{Sign}(\text{sk}, m)$ with $m \in \{0, 1\}^n$, for every $\rho \in \{0, 1\}^n$, for every $\tilde{\text{vk}} = \text{MaulVK}(\text{vk}, \rho)$, for every $\tilde{\sigma} = \text{AlignSig}((\text{vk}, \sigma, m), \rho)$, the algorithm UnAlignSig returns the original signature σ , i.e., $\text{UnAlignSig}((\tilde{\text{vk}}, \tilde{\sigma}, m), \rho) = \sigma$

Definition 3 (Key-sanitizable EUF-CMA security). *A key-sanitizable signature scheme is key-sanitizable existentially unforgeable against chosen message attacks (KS-EUF-CMA) if the probability of the adversary \mathcal{A} winning the following game is negligible:*

- Sample $(\text{vk}, \text{sk}) \leftarrow_{\S} \text{KeyGen}(1^\lambda)$ and a blinding factor $\rho \leftarrow_{\S} \{0, 1\}^n$, and run $\mathcal{A}(\text{vk}, \rho)$. Compute $\tilde{\text{vk}} = \text{MaulVK}(\text{vk}, \rho)$.
- Upon receiving a query from \mathcal{A} with message m , compute $\sigma = \text{Sign}(\text{sk}, m)$ and $\text{AlignSig}((\text{vk}, \sigma, m), \rho)$. Respond with $\tilde{\sigma}$ and add m to a list \mathcal{M} .
- Challenge \mathcal{A} to produce a signature $\tilde{\sigma}^*$ on message $m^* \notin \mathcal{M}$ that verifies under $\tilde{\text{vk}}$.
- Upon receiving a response $(m^*, \tilde{\sigma}^*)$, \mathcal{A} wins if $\text{Vrfy}_{\tilde{\text{vk}}}(m^*, \tilde{\sigma}^*) = 1$.

Lemma 1. *Any EUF-CMA signature scheme that supports algorithms MaulVK , AlignSig , and UnAlignSig , as defined in Definition 2, is also KS-EUF-CMA.*

The proof consists of a black-box reduction to EUF-CMA, and is deferred to the full version.

Combining Verification Keys. Looking ahead, the link initialization phase of the protocol realizing $s\mathcal{F}_{\text{SAT}}$ relies on the determination of session IDs via (identifying) verification keys of parties, which get sanitized by firewalls in different directions. For instance, in the 2-party setting, core C_i has access to vk_i and $\tilde{\text{vk}}_j$, and core C_j has access to $\tilde{\text{vk}}_i$ and vk_j , with $\tilde{\text{vk}}_i, \tilde{\text{vk}}_j$ being appropriate sanitations of vk_i, vk_j using the same randomness ρ_i . For this reason, we additionally define an appropriate generic algorithm that allows to combine these keys either way to output the same value.

Definition 4 (Consistent identity combinability). *A key-sanitizable signature scheme has consistent identity combinability if it supports an algorithm IDComb with the following property:*

$$\text{IDComb}(\text{vk}_i, \text{MaulVK}(\text{vk}_j, \rho)) = \text{IDComb}(\text{MaulVK}(\text{vk}_i, \rho), \text{vk}_j).$$

Instantiation from BLS. We report the BLS signature scheme [10] in the following.

- $\text{KeyGen}(1^\lambda) = (\text{sk}, \text{vk}) = (x, g^x)$
- $\text{Sign}(\text{sk}, m) = H(m)^{\text{sk}}$
- $\text{Vrfy}(\text{vk}, (m, \sigma))$: Check $\hat{e}(\sigma, g) = \hat{e}(H(m), \text{vk})$

The BLS signature scheme is already compatible with all the additional interfaces required by a key-sanitizable signature scheme. Moreover, bilinear maps immediately induce the consistent identity combinability property:

- $\text{MaulVK}(\text{vk}, \rho) = \text{vk}^\rho$
- $\text{AlignSig}((\text{vk}, \sigma, m), \rho) = \sigma^\rho$
- $\text{UnAlignSig}((\text{vk}, \tilde{\sigma}, m), \rho) = \tilde{\sigma}^{\rho^{-1}}$
- $\text{IDComb}(\text{vk}_i, \text{vk}_j) = \hat{e}(\text{vk}_i, \text{vk}_j)$

Theorem 5. *The BLS signature scheme [10] with the additional algorithms specified above is a key-sanitizable signature scheme with KS-EUF-CMA security and consistent identity combinability, assuming that H is a random oracle and that CDH is hard for \mathbb{G} .*

The theorem follows by inspecting the newly-introduced algorithms, and by observing that the BLS signature scheme is EUF-CMA. We defer the formal proof to the full version.

Realizing $s\mathcal{F}_{\text{SAT}}$. We now describe a protocol that realizes $s\mathcal{F}_{\text{SAT}}$ in the 2-party setting, which follows a similar structure to that of [7, Section 4.2]. The link initialization phase is depicted in Fig. 5, and the message authentication phase in Fig. 6. A verbose description of the protocol can be found in the full version.

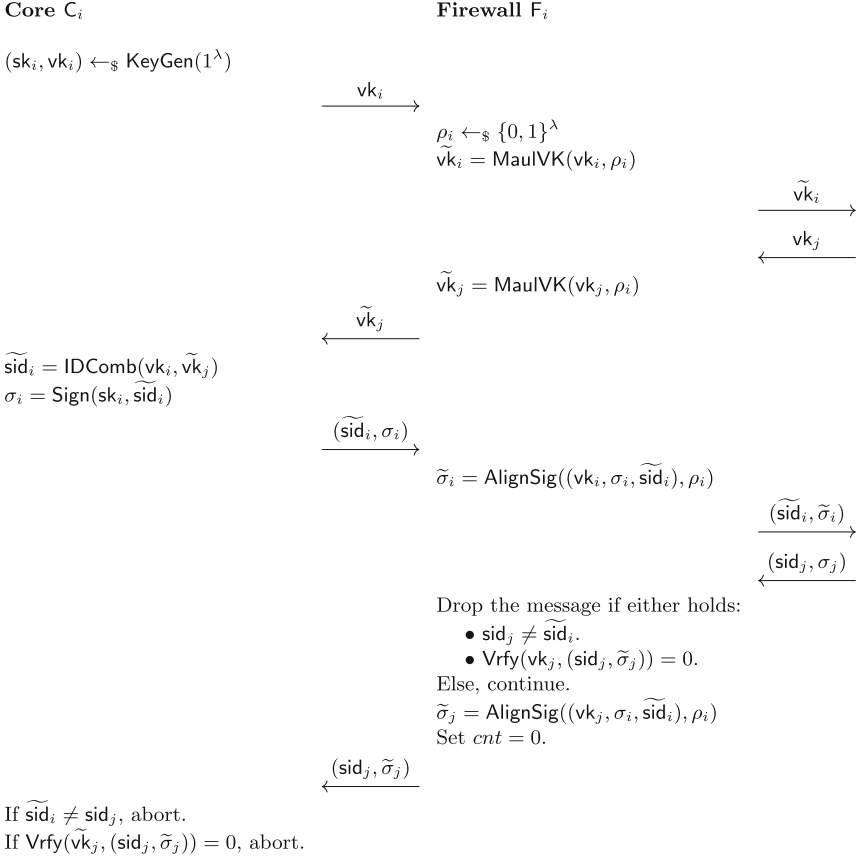


Fig. 5. Diagram of the protocol implementing the link initialization phase of $s\mathcal{F}_{\text{SAT}}$.

Theorem 6. *The protocol depicted in Figs. 5, 6 realizes the $s\mathcal{F}_{\text{SAT}}$ functionality, assuming a KS-EUF-CMA signature scheme with consistent identity combinability and the presence of secure channels between cores and their respective firewall.*

Intuitively, the proof runs as the one for the non-sanitized protocol of [7], except that the blinding operations of firewalls thwart subversion attacks, and consistency between keys is obtained by using IDComb. We defer the formal proof to the full version.

5.2 Multi-realizing any Ideal Functionality in the wsrUC Model

Next, we prove the following lemma.

Lemma 2. *For any regular (well-formed) ideal functionality \mathcal{F} there exists a protocol π that n -realizes \mathcal{F} in the wsrUC model assuming authenticated channels*

in the presence of static and malicious adversaries for $n = \text{poly}(\lambda)$. Moreover, the protocol π is such that all instances of π use a single instance of \mathcal{F}_{crs} .

Informally, such a protocol can be obtained from the adaptation of the GMW compiler to the srUC framework shown in [18]. The formal proof of the lemma is essentially [7, Theorem 13] verbatim, except that we replace results for the UC framework with their counterparts in the srUC framework, shown in [18] (e.g., the UC composition theorem and the GMW compiler). We defer the formal proof to the full version.

5.3 Realizing Generic Split Functionalities

We finally show that any protocol π that wsrUC-2-realizes a 2-party functionality \mathcal{F} in the \mathcal{F}_{SAT} -hybrid model (i.e., using authenticated channels) can be compiled into a protocol Π that wsrUC-realizes the split 2-party functionality $s\mathcal{F}$ in the $s\mathcal{F}_{\text{SAT}}$ -hybrid model (i.e., using unauthenticated channels). The $s\mathcal{F}$ functionality is exactly the same as in [7]. Indeed, since we wsrUC-realize a *regular* ideal functionality \mathcal{F} assuming \mathcal{F}_{SAT} , our end goal is to wsrUC-realize the split counterpart of \mathcal{F} assuming $s\mathcal{F}_{\text{SAT}}$, which is also a *regular* ideal functionality.

Lemma 3. *Let \mathcal{G} be a setup functionality, let \mathcal{F} be a 2-party ideal functionality, and let $\pi_{\mathcal{F}}$ be a protocol that securely 2-realizes \mathcal{F} in the wsrUC model with*

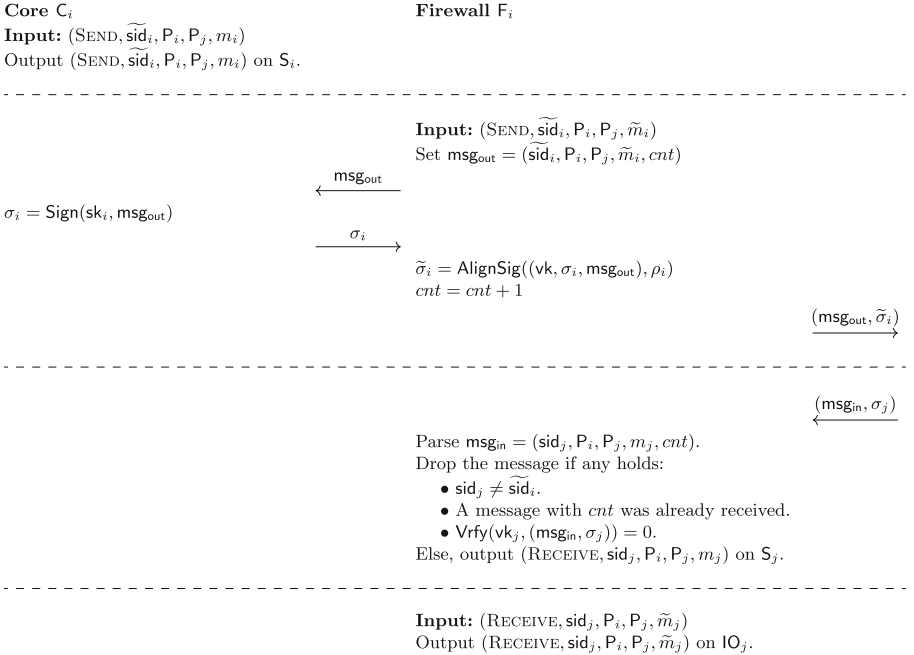


Fig. 6. Diagram of the protocol implementing the message authentication $s\mathcal{F}_{\text{SAT}}$, split in each of the interfaces.

authenticated communication (i.e., \mathcal{F}_{SAT}) and a single instance of \mathcal{G} . Then, there exists a protocol $\Pi_{\mathcal{F}}$ wsrUC-realizing the split functionality $s\mathcal{F}$ using a single instance of $s\mathcal{F}_{\text{SAT}}$ and a single instance of \mathcal{G} .

To prove this theorem, we adapt the proof of [7, Lemma 4.1] to the wsrUC model. First, we describe the protocol $\Pi_{\mathcal{F}}$, which is obtained by adapting the compiler presented in [7]. In particular, the compiler of [7] transforms a protocol $\pi_{\mathcal{F}}$ realizing functionality \mathcal{F} in the UC $\mathcal{F}_{\text{MAUTH}}$ -hybrid model into a protocol $\Pi_{\mathcal{F}}$ realizing functionality $s\mathcal{F}$ in the UC \mathcal{F}_{SA} -hybrid model. This result can be mapped to our setting by replacing $\mathcal{F}_{\text{MAUTH}}$ with \mathcal{F}_{SAT} , and \mathcal{F}_{SA} with $s\mathcal{F}_{\text{SAT}}$, with the crucial detail that messages coming from $s\mathcal{F}_{\text{SAT}}$ are forwarded to the instance of the protocol $\pi_{\mathcal{F}}$ on the respective interface (i.e., IO or S), rather than having a single interface for each party. Then, we simply follow the proof of [7, Lemma 4.1] accounting for the additional communication between cores and firewalls and for the presence of specious cores, as per the srUC framework. We defer the description of $\Pi_{\mathcal{F}}$ and the formal proof to the full version.

Putting it All Together. We showed that the split functionalities notion of [7] can be cast in the subversion-resilient UC model in the same way as in standard UC. Namely, one can build a protocol n -realizing a functionality for the authenticated channel setting and simply invoke Lemma 3 to obtain security of the split version of the protocol in the unauthenticated channel setting (albeit only for 2-party functionalities). Since there exists a protocol 2-realizing *any* regular ideal functionality in the authenticated setting (by using the srUC GMW compiler of [18], as per Lemma 2), there also exists a matching 2-party protocol in the unauthenticated setting realizing the split version of the same functionality, yielding Theorem 4.

6 Sanitizing PAKE

So far we have only referred to the \mathcal{F}_{RE} functionality, in which the adversary is unable to perform any (online) password guesses. In order to move to PAKE, we first provide a description of $\mathcal{F}_{\text{PAKE}}$, highlighting its differences with respect to \mathcal{F}_{RE} . Then, similarly to [12], we argue that our protocol in Sect. 4 can be compiled in a protocol for $s\mathcal{F}_{\text{RE}}$ by invoking a result of Sect. 5. Finally, we show that $s\mathcal{F}_{\text{RE}}$ is sufficient to trivially realize $\mathcal{F}_{\text{PAKE}}$. We conclude the section by highlighting that it is also possible to obtain a protocol for $s\mathcal{F}_{\text{RE}}$ by using the general-purpose result given by Theorem 4 (which internally relies on the srUC GMW compiler). In that regard, we provide a hand-wavy performance comparison of such a protocol with our instantiation from DDH.

6.1 Description of $\mathcal{F}_{\text{PAKE}}$

The behaviour of $\mathcal{F}_{\text{PAKE}}$ is conceptually close to that of the \mathcal{F}_{RE} we described in Sect. 4.1, with the important difference that the adversary is now allowed

to perform (online) password guesses in order to influence the keys output by the functionality. In what follows, we provide a formal description of the $\mathcal{F}_{\text{PAKE}}$ functionality [12] that embeds minor variations to achieve consistency with \mathcal{F}_{RE} , and technical improvements from Dupont *et al.* [22].

Functionality $\mathcal{F}_{\text{PAKE}}$

The functionality $\mathcal{F}_{\text{PAKE}}$ is parameterized by a security parameter λ , an initiator I , a responder R , and the adversary \mathcal{S} via the following queries:

Upon receiving a query (NEWSESSION, sid, I, R, w^I) **from** I :

Record (I, R, w^I) , mark it as **fresh**, and leak (sid, I, R) to \mathcal{S} . Ignore all future messages from I .

Upon receiving a query (OK, sid) **from** \mathcal{S} :

Send a message (WAKEUP, sid, I, R) to R . Ignore all future (OK) messages.

Upon receiving a query (RESPOND, sid, I, R, w^R) **from** R :

Record (R, I, w^R) and mark it as **fresh**.

Upon receiving a query (TESTPWD, sid, P, w') **from the adversary** \mathcal{S} :

If $P \in \{I, R\}$ and there exists a record of the form (P, \cdot, w) which is **fresh**, then:

- If $w' = w$, mark the record as **compromised** and return "correct guess" to \mathcal{S} .
- If $w' \neq w$, mark the record as **interrupted** and return "wrong guess" to \mathcal{S} .

Upon receiving a query (NEWKEY, sid, P_i, K) **from** \mathcal{S} , **where** $|K| = \lambda$:

If $P_i \in \{I, R\}$ and there is a record of the form (P_i, P_j, w_i) that is not marked as **completed**, with P_j being the peer of P_i , then:

- If any of the following conditions hold, output (sid, K) to party P_i :
 - P_i is corrupted.
 - This record is **fresh**, there exists a record (P_j, P_i, w_j) with $w_i = w_j$, and P_j is corrupted.
 - This record is **compromised**.
- If this record is **fresh**, both parties are honest, and there exists a record (P_j, P_i, w_j) with $w_j = w_i$, choose $\text{key} \leftarrow_{\mathcal{S}} \{0, 1\}^\lambda$. Output key to P_i , and append key to the record (P_i, P_j, w_i) .
- If this record is **fresh**, both parties are honest, and there exists a record $(P_j, P_i, w_j, \text{key})$ with $w_j = w_i$, output key to P_i .
- If none of the above rules apply, choose $\text{key}' \leftarrow_{\mathcal{S}} \{0, 1\}^\lambda$ and output it to party P_i .

In any case, mark the record (P_i, \cdot, w_i) as **completed**.

Variations in the srUC Setting. As for \mathcal{F}_{RE} , we restrict our attention to implicit mutual authentication (as discussed in Sect. 1.3), and the functionality provides no security whatsoever whenever the adversary is able to guess an honest party's password.

Shortcomings of PAKE Functionalities. Recent works have raised technical concerns regarding the definition of PAKE functionalities widely used across the literature. Specifically, Abdalla *et al.* [1] observed that several definitions, including the one of the seminal paper of Canetti *et al.* [13], allow the adversary to set the key output by an honest party even without knowing the password. Similarly to Dupont *et al.* [22], our definitions of \mathcal{F}_{RE} and $\mathcal{F}_{\text{PAKE}}$ do not embed this shortcoming.

Additionally, Roy and Xu [30] show an impossibility result proving that any 2-party $\mathcal{F}_{\text{PAKE}}$ may be instantiated by an incorrect 0-round protocol. In order to overcome this limitation, they show that either (i) the underlying PAKE protocol is assumed to be correct; (ii) the simulator gets limited in power; or (iii) a third party responsible for routing messages is introduced in $\mathcal{F}_{\text{PAKE}}$. For this work, we solve this shortcoming by considering approach (i), following the spirit of discarding “trivial protocols” in the context of UC (*e.g.*, the empty protocol), as discussed by Canetti *et al.* [14].

6.2 From \mathcal{F}_{RE} to $\mathcal{F}_{\text{PAKE}}$

The protocol we presented in Sect. 4 realizes \mathcal{F}_{RE} in the presence of subversion attacks in the authenticated setting. Proceeding as [12], we convert it to a protocol for $s\mathcal{F}_{\text{RE}}$, obtaining the following theorem:

Theorem 7. *There exists a protocol that wsrUC -realizes the $s\mathcal{F}_{\text{RE}}$ ideal functionality in the $(\mathcal{F}_{\text{crs}}, s\mathcal{F}_{\text{SAT}})$ -hybrid model under static corruptions. The protocol is based on the DDH assumption, runs in a constant number of rounds, and has a communication complexity of $O(n)$ group elements per session key.*

Proof (Theorem 7). The proof of this theorem is the proof of [12, Theorem 2] verbatim. First, we observe that the multi-session version of \mathcal{F}_{RE} can be implemented by having access to the multi-session version of \mathcal{F}_{sOT} (each new session of \mathcal{F}_{RE} uses a new invocation of the protocol for \mathcal{F}_{sOT}). Then, we observe that our protocol in Sect. 3 implements the multi-session version of \mathcal{F}_{sOT} in the \mathcal{F}_{crs} -hybrid model. Hence, we can invoke Lemma 3, which allows us to replace \mathcal{F}_{SAT} with $s\mathcal{F}_{\text{SAT}}$, yielding a protocol for the split version of randomized equality (*i.e.*, $s\mathcal{F}_{\text{RE}}$).

All that remains to show is that $\mathcal{F}_{\text{PAKE}}$ can be instantiated from $s\mathcal{F}_{\text{RE}}$. Intuitively, the power of the adversary to disconnect parties in $s\mathcal{F}_{\text{RE}}$ can be mapped to TESTPWD queries in $\mathcal{F}_{\text{PAKE}}$, as the adversary is allowed to run \mathcal{F}_{RE} with an arbitrary password by impersonating a disconnected party’s peer.

Theorem 8. *There exists a protocol in the $s\mathcal{F}_{\text{RE}}$ -hybrid model that instantiates $\mathcal{F}_{\text{PAKE}}$ in the presence of subversion attacks.*

Dupont *et al.* [22] exhibit a trivial protocol in the $s\mathcal{F}_{\text{RE}}$ -hybrid model that realizes $\mathcal{F}_{\text{PAKE}}$. In particular, their protocol exclusively interacts with $s\mathcal{F}_{\text{RE}}$. This fact allows to port their protocol and its related proof to our setting in a straightforward manner, as intuitively such a protocol inherits the structure and the security properties of $s\mathcal{F}_{\text{RE}}$. We report the formal proof in the full version.

6.3 A Hand-Wavy Performance Comparison

An alternative route to obtain $\mathcal{F}_{\text{PAKE}}$ consists of invoking Theorem 4 to obtain a protocol wsrUC -realizing $s\mathcal{F}_{\text{RE}}$, and then applying the transformation of Theorem 8. In particular, as per Lemma 2, this protocol relies on the srUC GMW compiler of [18]. In order to establish an informal comparison with our instantiation from DDH (given by Theorem 7), we first observe that both these protocols rely on Lemma 3 to move from the authenticated setting to the unauthenticated setting. Hence, it suffices to compare the protocols in the authenticated setting. For our hand-wavy comparison, we compare round complexity and communication complexity.

Our instantiation from DDH, as per Fig. 4, essentially relies on n runs of \mathcal{F}_{sOT} that share the same CRS. By our specific instantiation of \mathcal{F}_{sOT} , each party sends 1 public key and 2 SHDME encryptions (= 4 group elements) for each bit of the password. Hence, our protocol runs in 2 rounds (by batching messages for sOTs) with a communication complexity of $O(n)$ group elements.

On the other hand, the instantiation from the srUC GMW compiler requires each party to (i) generate its random tape jointly with its peer; (ii) commit to its input; (iii) prove in zero-knowledge that each step of a semi-honest protocol realizing \mathcal{F}_{RE} was executed correctly. (i) requires 3 rounds: 1 for committing to some locally-generated randomness and 2 from the coin tossing functionality. (ii) requires 1 round. (iii) requires at least the same number of rounds of a semi-honest execution of an r -round protocol realizing \mathcal{F}_{RE} . Hence, we end up with at least $4 + r$ rounds. We then observe that the coin tossing functionality of [18, Section 4] relies on the sanitizable commitment functionality (presented in [18, Section 3]), which is realized by computing and forwarding bit-wise commitments (each containing 2 group elements) under the DDH assumption. Given that the input to the semi-honest instantiation of \mathcal{F}_{RE} is an n -bit password, and that the random strings used to generate the random tape have size λ , the communication complexity of the first two steps is already $O(n + \lambda)$.

We conclude that our instantiation from DDH has a better round and communication complexity even prior to the run of the compiled semi-honest instantiation of \mathcal{F}_{RE} of the protocol from GMW. We further remark that, in step (iii), the protocol from GMW requires the generation of re-randomizable NIZK arguments for each message of the protocol, hindering the efficiency further.

7 Conclusions

We presented the first subversion-resilient UC protocol for PAKE. We formalized and instantiated oblivious transfer in the subversion setting, and extended the framework to the unauthenticated setting, providing an implementation for its respective backbone of communication (*i.e.*, $s\mathcal{F}_{\text{SAT}}$) in the two-tier model without assuming a PKI. Finally, we instantiated $\mathcal{F}_{\text{PAKE}}$ by replacing, in a sanitized protocol for \mathcal{F}_{RE} , the \mathcal{F}_{SAT} assumption with $s\mathcal{F}_{\text{SAT}}$. Several interesting research questions remain open, such as fully instantiating \mathcal{F}_{SAT} in the two-tier model, expanding the notion of split functionalities in the srUC model to the n -party setting, extending the framework to adaptive corruptions, weakening

trusted setups to be subvertable, and achieving explicit mutual authentication for randomized equality and PAKE.

References

1. Michel Abdalla, Björn Haase, and Julia Hesse. Security analysis of CPlace. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 711–741. Springer, Cham, December 2021.
2. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, and Michal Zajac. A subversion-resistant SNARK. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part III*, volume 10626 of *LNCS*, pages 3–33. Springer, Cham, December 2017.
3. Behzad Abdolmaleki, Helger Lipmaa, Janno Siim, and Michal Zajac. On QA-NIZK in the BPK model. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 590–620. Springer, Cham, May 2020.
4. Paula Arnold, Sebastian Berndt, Jörn Müller-Quade, and Astrid Ottenhues. Protection against subversion corruptions via reverse firewalls in the plain universal composability framework. Cryptology ePrint Archive, Report 2023/1951, 2023.
5. Giuseppe Ateniese, Danilo Francati, Bernardo Magri, and Daniele Venturi. Public immunization against complete subversion without random oracles. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19 International Conference on Applied Cryptography and Network Security*, volume 11464 of *LNCS*, pages 465–485. Springer, Cham, June 2019.
6. Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. Subversion-resilient signature schemes. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *ACM CCS 2015*, pages 364–375. ACM Press, October 2015.
7. Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 361–377. Springer, Berlin, Heidelberg, August 2005.
8. Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. NIZKs with an untrusted CRS: Security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 777–804. Springer, Berlin, Heidelberg, December 2016.
9. Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. Security of symmetric encryption against mass surveillance. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 1–19. Springer, Berlin, Heidelberg, August 2014.
10. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Berlin, Heidelberg, December 2001.
11. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.
12. Ran Canetti, Dana Dachman-Soled, Vinod Vaikuntanathan, and Hoeteck Wee. Efficient password authenticated key exchange via oblivious transfer. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 449–466. Springer, Berlin, Heidelberg, May 2012.

13. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, Berlin, Heidelberg, May 2005.
14. Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 68–86. Springer, Berlin, Heidelberg, May 2003.
15. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.
16. Suvradip Chakraborty, Stefan Dziembowski, and Jesper Buus Nielsen. Reverse firewalls for actively secure MPCs. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 732–762. Springer, Cham, August 2020.
17. Suvradip Chakraborty, Chaya Ganesh, Mahak Pancholi, and Pratik Sarkar. Reverse firewalls for adaptively secure MPC without setup. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 335–364. Springer, Cham, December 2021.
18. Suvradip Chakraborty, Bernardo Magri, Jesper Buus Nielsen, and Daniele Venturi. Universally composable subversion-resilient cryptography. In Orr Dunkelman and Stefan Dziembowski, editors, *EUROCRYPT 2022, Part I*, volume 13275 of *LNCS*, pages 272–302. Springer, Cham, May / June 2022.

19. Rongmao Chen, Yi Mu, Guomin Yang, Willy Susilo, Fuchun Guo, and Mingwu Zhang. Cryptographic reverse firewall via malleable smooth projective hash functions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part I*, volume 10031 of *LNCS*, pages 844–876. Springer, Berlin, Heidelberg, December 2016.
20. Jean Paul Degabriele, Pooya Farshim, and Bertram Poettering. A more cautious approach to security against mass surveillance. In Gregor Leander, editor, *FSE 2015*, volume 9054 of *LNCS*, pages 579–598. Springer, Berlin, Heidelberg, March 2015.
21. Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. Message transmission with reverse firewalls—secure communication on corrupted machines. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 341–372. Springer, Berlin, Heidelberg, August 2016.
22. Pierre-Alain Dupont, Julia Hesse, David Pointcheval, Leonid Reyzin, and Sophia Yakubov. Fuzzy password-authenticated key exchange. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part III*, volume 10822 of *LNCS*, pages 393–424. Springer, Cham, April / May 2018.
23. Georg Fuchsbauer. Subversion-zero-knowledge SNARKs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 315–347. Springer, Cham, March 2018.
24. Chaya Ganesh, Bernardo Magri, and Daniele Venturi. Cryptographic reverse firewalls for interactive proof systems. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *ICALP 2020*, volume 168 of *LIPICs*, pages 55:1–55:16. Schloss Dagstuhl, July 2020.
25. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.
26. Adam Groce and Jonathan Katz. A new framework for password-based authenticated key exchange. Cryptology ePrint Archive, Report 2010/147, 2010.
27. Ilya Mironov and Noah Stephens-Davidowitz. Cryptographic reverse firewalls. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 657–686. Springer, Berlin, Heidelberg, April 2015.
28. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Berlin, Heidelberg, August 2008.
29. Magnus Ringerud. Note on subversion-resilient key exchange. Cryptology ePrint Archive, Report 2023/749, 2023.
30. Lawrence Roy and Jiayu Xu. A universally composable PAKE with zero communication cost - (and why it shouldn't be considered UC-secure). In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 714–743. Springer, Cham, May 2023.
31. Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Cliptography: Clipping the power of kleptographic attacks. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 34–64. Springer, Berlin, Heidelberg, December 2016.
32. Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. Generic semantic security against a kleptographic adversary. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 907–922. ACM Press, October / November 2017.

33. Gustavus J. Simmons. Authentication theory/coding theory. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 411–431. Springer, Berlin, Heidelberg, August 1984.
34. Gustavus J. Simmons. A secure subliminal channel (?). In Hugh C. Williams, editor, *CRYPTO'85*, volume 218 of *LNCS*, pages 33–41. Springer, Berlin, Heidelberg, August 1986.



Tightly-Secure Group Key Exchange with Perfect Forward Secrecy

Emanuele Di Giandomenico¹, Doreen Riepel², and Sven Schäge¹

¹ Eindhoven University of Technology, Eindhoven, Netherlands
{e.di.giandomenico,s.schage}@tue.nl

² CISPA Helmholtz Center for Information Security, Saarbrücken, Germany

Abstract. In this work, we present a new paradigm for constructing Group Authenticated Key Exchange (GAKE). This result is the first tightly secure GAKE scheme in a strong security model that allows maximum exposure attacks (MEX) where the attacker is allowed to either reveal the secret session state or the long-term secret of all communication partners. Moreover, our protocol features the strong and realistic notion of (full) perfect forward secrecy (PFS), that allows the attacker to actively modify messages before corrupting parties. We obtain our results via a series of tightly secure transformations. Our first transformation is from weakly secure KEMs to unilateral authenticated key exchange (UAKE) with weak forward secrecy (WFS). Next, we show how to turn this into an UAKE with PFS in the random oracle model. Finally, and as one of our major novel conceptual contributions, we describe how to build GAKE protocols from UAKE protocols, also in the random oracle model. We apply our transformations to obtain two practical GAKE protocols with tight security. The first is based on the DDH assumption and features low message complexity. Our second result is based on the LWE assumption. In this way, we obtain the first GAKE protocol from a post-quantum assumption that is tightly secure in a strong model of security allowing MEX attacks.

1 Introduction

Group Authenticated Key Exchange (GAKE) is the generalization of two-party key exchange to the group setting. It allows N group members to compute a common symmetric session key over an insecure network. This key can then be used to exchange messages among the group members that are protected via efficient symmetric cryptography. As such GAKE protocols form an important building block in any form of group-based communication.

Proving security of GAKE protocols is in general much more challenging than for classical AKE protocols. In many AKE security proofs the two parties participating in the protocol can simply be guessed upfront resulting in a polynomial security loss $\binom{N}{2}$. For GAKE protocols this strategy quickly becomes infeasible with growing group size t since there are $\binom{N}{t}$ possible groups that could now run the GAKE protocol. For superlogarithmical t this number already grows

superpolynomial and guessing the group upfront becomes inefficient. Moreover, each of the existing GAKE protocols have one of the following downsides.

Vulnerability Against Quantum Attacks. The vast majority of GAKE protocols rely on classical security assumptions that are related to the discrete logarithm assumption. The underlying problems are known to be solvable efficiently by quantum computers [31]. For long-term security a shift towards post-quantum-based security assumptions is necessary. However, relying on post-quantum assumptions often introduces new challenges like non-perfect correctness in lattice cryptography. Thus PQ-based security assumptions cannot be used as a drop-in to classical protocols and new techniques are necessary.

Realistic Security Models. Most GAKE protocols consider relatively weak security definitions that only consider attackers that may corrupt the long-term keys of parties while disallowing that the (ephemeral) state material stored by parties between moves will ever be revealed. So, in case an attacker manages to obtain the state information of a *single* group member all security guarantees may be lost. A stronger and much more realistic notion of security considers so-called maximum exposure attacks (MEX) that allow the attacker to also reveal the secret states of the group members while carefully excluding trivial attacks [22]. These models are considered standard in the case of two-party key exchange.

Non-tight Security Proofs. A tight security proof allows for highly efficient and theoretically-sound instantiations of the system parameters. In particular, the proofs—and thus the system parameters—are independent of the number of parties, sessions, or the number of attacker queries. Providing schemes with tight security proofs for asymmetric cryptography is challenging [5], in particular for key exchange [8]. Only recently have tightly secure AKE protocols been proposed for strong security in the two-party case [15, 19, 28]. The only tightly secure GAKE protocol [25] relies on signatures (which are generally less efficient in the PQ-setting) and also does not protect against MEX attacks.

1.1 Contribution

In this work, we tackle these challenges and present the first, tightly secure GAKE protocol that is secure under post-quantum assumptions under a strong, realistic notion of security. To this end, we develop a new paradigm for constructing GAKE schemes. To explain it intuitively, consider the well-known ring-based Burmester-Desmedt (BD) protocol [7]. In this protocol, each party P_i first sends $k_i = g^{x_i}$ for some randomly drawn ephemeral secret x_i . In the next round, each party sends $K_i = (k_{i+1}/k_{i-1})^{x_i}$ where all indices are taken mod t for group size t . The final group key is produced as

$$K = k_{i-1}^{tx_i} K_i^{t-1} K_{i+1}^{t-2} \dots K_{i-2} = g^{x_1 x_2 + x_2 x_3 + \dots + x_t x_1}.$$

This protocol is only passively secure but serves as a guiding principle in many constructions. Using digital signatures over all messages sent, this protocol can be made actively secure (though it remains highly vulnerable to state-reveal attacks). While it is elegant, we believe that it rather disguises the core principles that make it work. We therefore present a more conceptual perspective to the design of GAKE protocols that to the best of our knowledge is novel. This allows us to identify the parts that can be improved considerably.

Novel Conceptual Perspective on GAKE. Assume we have t parties P_1, \dots, P_t organized in a ring. Essentially we view a GAKE protocol as consisting of two phases. In the first phase, adjacent parties compute a common session key via a two-party protocol. To make this secure against active attacks, the neighboring parties will at some point (implicitly or explicitly) authenticate each other. In particular, for each i , P_i authenticates P_{i-1} and P_{i+1} . In the basic BD protocol (which is only passively secure) this step simply consists of sending k_i . Actively secure protocols that rely on the BD protocol, typically add authentication via other means like digital signatures. More concretely, each party will also sign each message they send. The shared key with party P_{i+1} can then be computed as $G_{i,i+1} = (k_{i+1})^{x_i}$. Likewise, the shared key computed with P_{i-1} can then be computed as $G_{i-1,i} = (k_{i-1})^{x_i}$. The second phase of the protocol consists of distributing the derived key material to the other parties. Now, a key insight is that, in order to not hand these keys over to an impersonating attacker, they are only given to parties that P_i has authenticated before. In particular, $G_{i,i+1}$ is only given to P_{i-1} and $G_{i-1,i}$ is only given to P_{i+1} . In the BD protocol this is done simultaneously via simply publishing $G_{i,i+1}/G_{i-1,i}$. This can be thought as a simple symmetric encryption of $G_{i,i+1}$ (respectively $G_{i-1,i}$) via the key $G_{i-1,i}$ (respectively $G_{i,i+1}$). Now observe that from the knowledge of all the K_j , each party P_i can now easily compute $K = g^{x_1x_2+x_2x_3+\dots+x_tx_1} = G_{1,2}G_{2,3}\dots G_{t,1}$. It first computes $G_{i,i+1}$ using k_{i+1} and x_i . Next, it can step-wisely compute the next value $G_{i+1+c,i+2+c}$ from K_{i+1+c} and $G_{i+c,i+1+c}$ for any $c = 0, 1, \dots, t-1$. The BD protocol essentially computes this process in an algebraically elegant and efficient fashion.

Having this perspective in mind, we make several conceptual changes to the design that enable better efficiency. First, we do not require that P_i authenticates P_{i+1} and vice versa. Crucially, we observe that only one direction is enough. This is because the group members are organized in a ring: if each member authenticates its predecessor only, all parties will be authenticated eventually. In general, reciprocal authentication among neighbors seems wasteful. In addition to that, this change will now allow us to avoid using AKE protocols but instead rely on unilaterally authenticated key exchange (UAKE) where only a single party is authenticated. Overall this saves bandwidth and computational complexity. Let us clarify: each party will as before compute two shared keys one with its predecessor and one with its successor. However, only the predecessor will be authenticated. Thus, in the second phase, parties will now only distribute the symmetric key that they share with their successor to their authenticated predecessor (and not vice versa). The second change that we make is that we consider

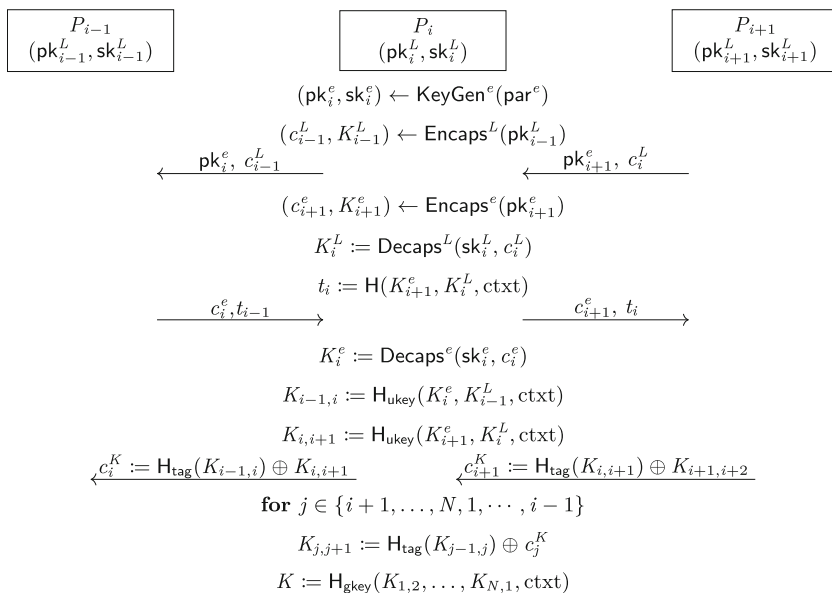


Fig. 1. The idea of the complete construction presented in Sect. 6, assuming, without loss of generality, that the predecessor and the successor of party i are $i-1, i+1$, respectively, and that the parties verify the tag received. The superscripts e and L stand for “ephemeral” and “long-term” respectively. H_* with different subscripts $*$ are (independent) random oracles.

the symmetric encryption scheme used more generally in the second phase. To this end, we use a simple random oracle-based symmetric encryption system, where the sharing of key $K_{i,i+1}$ to P_i now proceeds as $h(K_{i-1,i}) \oplus K_{i,i+1}$. In this way, each party essentially only encrypts to one party – its predecessor. This scheme is very simple and fast and has strong security properties. Unfortunately, currently there is no efficient and tightly secure post-quantum secure digital signature scheme that is suitable to implementing authentication efficiently.¹ To obtain a tightly secure GAKE protocol overall—even in the PQ-setting—we will thus deviate from the use of digital signature schemes and instead rely on authentication via KEMs, similar to previous work on tight AKE in the 2-party setting [19, 26, 28]. We provide an illustration of our protocol from the view of one party in Fig. 1. Typically post-quantum signatures are considerably larger in size than KEM ciphertexts. Essentially, the mechanism that we use for authentication will require P_i to send an encapsulated key to its predecessor (encrypted with pk_{i-1}). This key will be decapsulated by P_{i-1} and is then used to derive a MAC key, which in turn is used to provide integrity protection for all the messages sent and received by P_{i-1} . Since P_i knows the encapsulated key as

¹ The signature schemes introduced in [16, 27] do provide (almost) tight security but are too inefficient for practical applications.

well, it can recompute the MAC. The security properties of the KEM guarantee that the MAC can only be computed correctly by the predecessor if it indeed has the corresponding secret long-term key. In our instantiations, we rely on the recent tightly secure KEM from [28] that is secure under lattice assumptions and the DDH based scheme introduced in [19] that both fulfill the notion of OW-PCVA-CR [26], a very weak notion of KEM security.

We proceed as follows. First, we present a construction of an UAE scheme with weak perfect forward secrecy (WFS) that is constructed from a KEM. Next, we present a transformation from a WFS-secure UAE to an UAE that provides full PFS in the random oracle model (ROM). Whereas WFS only provides security guarantees against long-term key corruptions in case the attacker has *not* modified the sent messages, full PFS also guarantees security in the presence of active attackers that modify messages. Finally, and as our main contribution, we present a transformation from PFS-secure UAE to PFS-secure GAKE, again in the ROM. Security holds even under MEX attacks where the attacker may adaptively reveal state information and adaptively corrupt parties. Remarkably, all our transformations preserve the tightness of the security proof so that the final GAKE will tightly reduce to the security of the KEM. When instantiated with the PQ-secure scheme of [28] this results in the first tightly secure GAKE scheme under lattice assumptions in a very strong model of security. When instantiated under the DDH assumption, our protocol only requires to send 5 group elements and two bitstrings of length 256 bits per party. In comparison, the tightly secure protocol of [25] requires to send 2 group elements and 4 exponents (when relying on generic group model bounds of Schnorr signatures), or 2 group elements and 6 exponents (when instantiating the signature scheme with [10]).

1.2 Security Model

To provide a strong notion of security that reflects full PFS and security against state-reveal attacks, we present a new security definition. We remark that providing security notions for GAKE has in the past proven error-prone. This is due to the number of subcases that one has to consider in the proof. In this work, we take a new avenue that simplifies the development of such a definition.

The central idea is to strongly rely on a corresponding security definition for two parties. This definition is now used more generically to develop the GAKE definition. To this end, we take the strong definition of [19] as a starting point. This definition features an attack table that defines when certain query combinations of the attacker are deemed non-trivial. In our new definition a similar attack table is (almost) generically utilized at the end of our security experiment to evaluate if, for any of the tested sessions, the attacker has performed a trivial attack. However, we need to be careful since the application of the checks in the [19] table do not only depend on the holder of the tested session itself, but also on its peer.

Our formulation of GAKE security thus essentially re-applies this table to all the peers of the considered tested session that are currently participating in the GAKE run. In this way, we can reduce the problem of analyzing trivial

attacks for a session and all its peers to the problem of analyzing trivial attacks for this session and a single peer. We note that the semantics of these tables define when an attack is valid. In this sense they encode properties of non-trivial attacks. By setup all other attacks are deemed trivial. So to make this useful in the group setting we require that for all of the pairs of parties, the attacker hasn't performed a trivial attack.

We note that our GAKE definition holds for any polynomial-sized groups, in particular for groups of size 2. This implies a definition for classical AKE as well. However, conceptually the exposition of our algorithms is structured into rounds, where every party has to apply the same algorithms. This allows us to specify algorithms independent of classical roles like initiator or responder. However, when proving the security of our GAKE from the underlying UAKE we have to relate the UAKE roles of initiator and responder to the behaviour of two adjacent parties in the GAKE using terms like predecessor and successor.

$$\text{KEM} \xrightarrow{\text{Theorem 3}} \text{UAKE}_{\text{WFS}} \xrightarrow{\text{Theorem 1}} \text{UAKE}_{\text{PFS}} \xrightarrow{\text{Theorem 2}} \text{GAKE}_{\text{PFS}}$$

Fig. 2. Logical implication sequence from KEM to GAKE_{PFS} with intermediate steps.

1.3 Related Work

There has been considerably less research activity on GAKE than on classical two-party AKE. A nice overview of the existing notions of group AKE can be found in [30].

The protocol from [12] is similar to BD. It relies on DDH and signatures to achieve PFS. It supports dynamic groups and only requires two rounds. We note that while we require three rounds to get PFS, it was shown in [21] for 2-party AKE that if the underlying protocol is only implicitly authenticated (e.g., via KEMs), then a protocol cannot achieve PFS in two rounds.

The protocol in [4] can be thought of as a lattice-based variant of the BD protocol that is secure under the Ring-LWE assumption. Correspondingly it is passively secure and needs additional authentication mechanisms for active security. To this end the authors propose the application of signatures. It is generally unclear how to do this in an efficient and tightly secure manner in the post-quantum setting. The security model of [4] does not allow the attacker to reveal secret state information.

The work in [25] focuses on tight security. It also takes the BD protocol as a basis and presents a tight proof in a security model that does not allow the attacker to reveal secret state information. The construction applies the efficient Schnorr signature scheme to protect the protocol against active attacks and achieve authentication. We deviate from these two approaches by considering tight security in strong models that allow MEX attacks. Moreover, we use a novel authentication mechanism that relies on KEMs instead of signatures. This allows

us to obtain efficient instantiations based on previous works. Our instantiation in the classical setting considers the highly efficient DDH-based scheme introduced in [19]. In the post-quantum setting we can apply the recent scheme of [28] that is based on the LWE assumption. However, from the description of the scheme in [28] it is not immediately clear if it can be applied to our transformation when used in the group setting. The problem is that the correctness of the scheme is only shown to hold with probability $(1 - z)$ where $z = \text{negl}(\kappa)$. This can be problematic when bounding the probability that *all* $N = \text{poly}(\kappa)$ KEM applications that are required in a run of the GAKE protocol provide correctness because $(1 - \text{negl}(\kappa))^N$ is only overwhelming if z is statistically small. Fortunately, we can show that z is indeed statistically small [28].

Recent works on AKE also aim at achieving tighter security reductions in the QROM [17, 26, 29]. The first AKE protocol proven secure in the QROM [17] suffers from a square-root security loss in the random oracle model. This was improved in [29] that provides a QROM proof with a loss only linear in the number of users. The resulting scheme only provides weak forward secrecy. Very recently, via an additional key confirmation move [26], this was lifted to a protocol that provides perfect forward secrecy, also with a linear loss in the number of users.

Another interesting work related to ours is the authentication compiler of Katz and Yung that constructs actively secure GAKE from a passively secure one. Essentially the paper proposes to authenticate all messages with digital signatures schemes as in the BD protocol. However, their analysis does not account for attacks that reveal ephemeral states. Also they do not specifically consider tight reductions. Our result, in contrast, uses authentication based on KEMs that provides efficient instantiations in the DH setting and the PQ-setting. At the same time, our solutions are tightly secure.

In 1999, Mayer and Yung have proposed a construction of GAKE from two-party AKE [24]. The model that they use is comparatively weak and does not consider attacks that reveal state information. Also, they rely on key exchange with mutual authentication that – when used in a ring setting – requires each party to be authenticated twice. Our solution based on UAKE, authenticates parties only once and is thus more efficient, while featuring tight security. Similarly, the work presented in [1] considers a compiler from AKE to GAKE. Again, the security model is weaker than ours and does not allow to reveal state information in the GAKE. As in [24] the compiler requires the computationally more complex notion of AKE whereas we solely require UAKE.

UAKE protocols and their security notions were previously studied in [11, 23], where the former proposes a 2-round forward-deniable and forward-secure UAKE from KEMs that is very similar to ours and the latter focuses on universal composability (UC) security. Further, [18] studies anonymity of UAKE. The main focus of these works is to study UAKE protocols themselves, whereas we use UAKE as a building block for GAKE. Hence, our security notion is tailored to be as weak as possible to enable our transformation, which makes it presumably weaker than (or incomparable to) the ones given in these works.

We mention that our notion of security covers key compromise impersonation (KCI) security for GAKE as introduced by [14]. Whereas [14] can be thought of as an analogue of the security notion introduced in [21], our notion rather generalizes the stronger notion of [22].

Finally, we remark that GAKE is generally related to Group Continuous Key Agreement, a notion that has gained much interest [2,3,9,20] in the last years. More formally, the authors of [6] provide initial results showing that weakly-secure variants of these primitives are indeed equivalent. We believe that this relationship will become much clearer in the future where we expect GAKE to be an essential primitive used in the setup phase of CGKA protocols to establish key material for the first time. It is thus very helpful that our protocol provides security even in case the attacker obtains secret state information. This seems helpful in CGKA constructions to achieve the intricate notions of post-compromise security that CGKA protocols try to guarantee in a provably secure way.

2 Preliminaries

For a positive integer N , let $[N] := \{1, \dots, N\}$. For a set S , let $|S|$ be the cardinality of S ; moreover, $s \leftarrow S$ denotes that s is sampled uniformly at random from S . We use the abbreviation $\llbracket B \rrbracket$ to represent the bit set to 1 when the boolean statement B is true, and 0 otherwise.

By $y \leftarrow \mathcal{A}(x)$, we denote that on input $x \in X$, the probabilistic algorithm \mathcal{A} returns $y \in Y$. Otherwise, by $y := \mathcal{A}(x)$, we denote that on input x , the deterministic algorithm \mathcal{A} returns y . By \mathcal{A}^O , we denote that the algorithm \mathcal{A} has access to oracle O . We say that probabilistic algorithm \mathcal{A} has min-entropy μ if for all outputs $y' \in Y$ we have $\Pr[y = y' : y \leftarrow \mathcal{A}(x)] \leq 2^{-\mu}$.

Following [32], we use code-based games. An adversary is a probabilistic polynomial time algorithm. Let G be a game, for an adversary \mathcal{A} , $G^{\mathcal{A}} \Rightarrow 1$ denotes that the output of game G running with adversary \mathcal{A} is 1. All the games that will be introduced later have two fixed oracles, INITIALIZE and FINALIZE, which can be queried at most once, as the first and last query respectively.

3 Unilateral Authenticated Key Exchange

We will first define unilateral authenticated key exchange (UAKE), which is a two-party protocol where only one party authenticates to the other. We will only focus on two-message protocols (but note that the syntax can be extended trivially).

SYNTAX. A two-message unilateral authenticated key exchange $\text{UAKE} := (\text{Setup}, \text{KeyGen}, \text{Beg}, \text{Der}_R, \text{Der}_B)$ consist of five polynomial-time algorithms:

- $\text{par} \leftarrow \text{Setup}(1^\kappa)$: The probabilistic setup algorithm Setup takes as input the security parameter κ in unary and returns global system parameters par that implicitly define message space \mathcal{T} , the public key space \mathcal{PK} , the secret key space \mathcal{SK} and the key space \mathcal{K} .

- $(pk, sk) \leftarrow \text{KeyGen}(\text{par})$: The probabilistic key generation algorithm KeyGen takes as input the parameters par and returns a public key $pk \in \mathcal{PK}$ and a secret key $sk \in \mathcal{SK}$.
- $(M_1, st) \leftarrow \text{Beg}(pk)$: The probabilistic initial algorithm Beg takes as input a public key pk and returns a message $M_1 \in \mathcal{T}$ and a state st .
- $(M_2, K) \leftarrow \text{Der}_R(sk, M_1)$: The probabilistic derivation for the responder algorithm Der_R takes as input a secret key sk and a message M_1 and returns a message $M_2 \in \mathcal{T}$ and a key $K \in \mathcal{K}$.
- $K := \text{Der}_B(pk, M_2, st)$: The deterministic derivation for the initiator algorithm Der_B takes as input a secret key pk , a message M_2 and a state st and returns a key $K \in \mathcal{K}$.

Note that only the party I save a state information, even if only the party R has long-term keys. Then, R can derive immediately the session key K after receiving the message of I (Fig. 3).

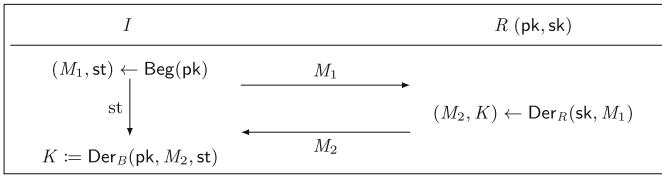


Fig. 3. Syntax of a two-message unilateral key exchange protocol.

Definition 1 (Correctness of UAE). We say that UAE is ρ -correct, if for any $\text{par} \leftarrow \text{Setup}(1^\kappa)$ we have:

$$\Pr \left[K = K' \mid \begin{array}{l} (pk, sk) \leftarrow \text{KeyGen}(\text{par}), \\ (M_1, st) \leftarrow \text{Beg}(pk), \\ (M_2, K) \leftarrow \text{Der}_R(sk, M_1), \\ K' := \text{Der}_B(pk, M_2, st) \end{array} \right] \geq \rho,$$

and the probability is over the random coins consumed by the algorithms of UAE.

Now we introduce the definition of min-entropy for UAE. This is extremely useful for the theorems we will introduce.

Definition 2 (Min-Entropy of UAE). We say that UAE has min-entropy μ if:

- It has key min-entropy $\mu' \geq \mu$: for any $pk' \in \mathcal{PK}$ we have $\Pr[pk = pk' : (pk, sk) \leftarrow \text{KeyGen}(\text{par})] \leq 2^{-\mu'}$ for some par .
- It has min-entropy $\mu'' \geq \mu$ of Beg : for any $M'_1 \in \mathcal{T}$ we have $\Pr[M_1 = M'_1 : (M_1, st) \leftarrow \text{Beg}(pk)] \leq 2^{-\mu''}$ for some $pk \in \mathcal{PK}$.
- It has min-entropy $\mu''' \geq \mu$ of Der_R : for any $M'_2 \in \mathcal{T}$ we have $\Pr[M_2 = M'_2 : (M_2, K) \leftarrow \text{Der}_R(sk, M_1)] \leq 2^{-\mu'''}$ for some $sk \in \mathcal{SK}$ and $M_1 \in \mathcal{T}$.

3.1 Security for UAKE

We consider N parties P_1, \dots, P_N (for an easier notation sometimes we use n to refer to P_n) with long-term key pairs $(\mathbf{pk}_n, \mathbf{sk}_n)$, $n \in [N]$. An interaction between two parties is called session, and to each session are associated an identification number sID and variables defined on sID .

- $\text{ini}[sID] \in [N]$ denotes the initiator of the session.
- $\text{res}[sID] \in [N]$ denotes the responder of the session.
- $\text{type}[sID] \in \{\text{“In”}, \text{“Re”}\}$ denotes if the initiator or the responder computes the session key.
- $I[sID]$ denotes the message sent by the initiator.
- $R[sID]$ denotes the message sent by the responder.
- $\text{state}[sID]$ denotes the state.
- $\text{sKey}[sID]$ denotes the session key. In case a party does not accept, this variable will be set to **rej**.

Moreover, we use the following boolean values to store which queries the adversary made.

- $\text{corr}[n]$ denotes if long-term secret key of party P_n has been given to the adversary.
- $\text{revState}[sID]$ denotes if the state has been given to the adversary.
- $\text{peerCorr}[sID]$ denotes if the peer of the session is corrupted and its long-term key has already been given to the adversary at the time the session key is derived.

Let us now define what it means for two sessions to be (partially) matching. As in two-party key exchange, the notion of partially matching sessions is used to define if two parties have communicated with each other and so revealing session secrets of the first will also reveal secrets of the second. Partially matching sessions take into account that parties have to accept at distinct points in time: the party responsible for sending the last message will accept independent of whether the attacker modifies the last message on transit or not. We will later use our general methodology and derive a definition of partially matching for GAKE protocols that is intuitively based on a repeated application of the two-party notion. This simplifies the exposition greatly.

Definition 3 (Partially Matching Session for UAKE). *The session sID is partially matched with session sID^* if the following conditions are satisfied.*

1. *The sessions have the same initiator and responder, $(\text{ini}[sID], \text{res}[sID]) = (\text{ini}[sID^*], \text{res}[sID^*])$.*
2. *The messages of the initiator are identical, $I[sID] = I[sID^*]$.*
3. *The types of the sessions are distinct, $\text{type}[sID] \neq \text{type}[sID^*]$.*
4. *The type of sID is “In”, $\text{type}[sID] = \text{“In”}$.*

Definition 4 (Matching Session for UAKE). *Two sessions sID and sID^* are matching if the following conditions are satisfied.*

1. *The sessions have the same initiator and responder, $(ini[sID], res[sID]) = (ini[sID^*], res[sID^*])$.*
2. *The messages of the initiator are identical, $I[sID] = I[sID^*]$.*
3. *The messages of the responder are identical, $R[sID] = R[sID^*]$.*
4. *The type of the sessions are distinct, $type[sID] \neq type[sID^*]$.*

OW-SECURITY. We define security in a one-way game, where the adversary has to compute the session key of a target session of its choice. The full game is given in Fig. 4. The adversary can create parties using the KEYGENERATION oracle. It can send and relay messages between the parties using oracles Beg , S_{Der_R} , S_{Der_B} . It can reveal the state of sessions and corrupt parties via REV-STATE and CORRUPT, respectively. Further, we allow the adversary to check for session keys using oracle CHECK. If the session is fresh when the oracle is queried and the key is correct, then we set a flag `attFound` which will also be the final output of the game, i.e., the adversary wins whenever this flag is set.

In order to rule out trivial attacks, we use attack tables as introduced in [19] to describe which queries the adversary is allowed to make for an attack. We give two tables, one capturing perfect forward secrecy (cf. Table 1) and the other one capturing weak forward secrecy (cf. Table 2) for UAKE protocols.

Each table is parameterized by an initiator i^* and responder r^* session. Note that only the responder has a public key, hence we only need to consider corruptions of that party. The initiator on the other hand holds a state, however, we do not allow the adversary to reveal the state for any session it wants to attack. Even though this could be achieved for some cases, we will see that it is not necessary to allow this attack for our following transformations. Similarly, we also omit a session key reveal oracle. This way, we relax the security definition as much as possible to allow for most efficient instantiations, while still achieving the strongest target notion for the final group AKE protocol.

We explain Table 1 in more detail, the other table can be read in a similar way. Line (0) captures that if a protocol has not sufficient entropy, then the protocol should not be considered secure. If this is the case, it should be possible for an adversary to create a session that has multiple (partially) matching sessions, so whenever this happens, we consider it a valid attack which lets the adversary win directly. Line (1) is for sessions that have a matching session. These can be of type “In” or “Re” and in this case we allow the adversary to reveal the responder’s long-term key, but (as explained above) we never allow to reveal the initiator’s state. Line (2) captures partially matching sessions which are always of type “Re”. For those, we also allow the responder to be corrupted. Line (3) captures sessions that do not have any (partially) matching partner session. Since Table 1 looks at perfect forward secrecy, we allow to reveal the responder’s secret key *after* the session key has been computed, which is captured by variable `peerCorr`. We only consider sessions of type “In” since the initiator has no long-term secrets and sessions of type “Re” can never be secure when being actively attacked.

For completeness, we give the table for weak forward security of UAKE protocols in Table 2. It is very similar to Table 1, the only difference is that the peer cannot be corrupted at all if the adversary was active.

GAMES OW-UAKE-G_X		$S_{Der_B}(sID, M_2)$
<u>INITIALIZE</u>		28 if state[sID] = \perp : return \perp
00 cnt := 0	//session counter	29 if sKey[sID] \neq \perp : return \perp
01 $N := 0$		30 $(i, r) := (\text{ini}[sID], \text{res}[sID])$
02 attFound := 0		31 peerCorr[sID] := corr[r]
03 $\text{par} \leftarrow \text{Setup}(1^\epsilon)$		32 st := state[sID]
04 return par		33 $K := \text{Der}_B(\text{pk}_r, M_2, \text{st})$
<u>KEYGENERATION</u>		34 $R[sID] := M_2$
05 $N ++$		35 sKey[sID] := K
06 $(\text{pk}_N, \text{sk}_N) \leftarrow \text{KeyGen}(\text{par})$		36 return ϵ
07 return pk_N		<u>REV-STATE(sID)</u>
<u>FINALIZE</u>		37 if type[sID] \neq "In" : return \perp
08 return attFound		38 revState[sID] := true
		39 return state[sID]
<u>S_{Beg}((i, r))</u>		<u>CORRUPT(n)</u>
09 if $(i, r) \notin [N]^2$: return \perp		40 corr[n] := true
10 cnt ++		41 return sk_n
11 sID := cnt		<u>CHECK(sID, K)</u>
12 $(\text{ini}[sID], \text{res}[sID]) := (i, r)$		42 if $K = \perp$: return \perp
13 type[sID] := "In"		43 if attFound = 0 :
14 $(M_1, \text{st}) \leftarrow \text{Beg}(\text{pk}_r)$		44 if sKey[sID] = K
15 $I[sID] := M_1$	//store initiator message	and UVALID _X (sID) = true :
16 state[sID] := st		45 $(s^*, k^*) := (sID, K)$
17 return (sID, M_1)		46 attFound := 1
		47 return [sKey[sID] = K]
<u>S_{Der_R}((i, r), M₁)</u>		
18 if $(i, r) \notin [N]^2$: return \perp		
19 cnt ++		
20 sID := cnt		
21 $(\text{ini}[sID], \text{res}[sID]) := (i, r)$		
22 type[sID] := "Re"		
23 $(M_2, K) \leftarrow \text{Der}_R(\text{sk}_r, M_1)$		
24 $I[sID] := M_1$		
25 $R[sID] := M_2$	//store responder message	
26 sKey[sID] := K		
27 return (sID, M_2)		

Fig. 4. Games OW-UAKE-G_X for a two-message UAKE. \mathcal{A} has access to oracles $\mathcal{O} := \{S_{\text{Beg}}, S_{\text{Der}_R}, S_{\text{Der}_B}, \text{REV-STATE}, \text{CORRUPT}, \text{CHECK}\}$. Helper procedure UVALID_X is defined in Fig. 5.

<u>UVALID_X(sID*)</u>	
00 $(i^*, r^*) := (\text{ini}[sID^*], \text{res}[sID^*])$	
01 $\mathfrak{M}(sID^*) := \{sID \mid (\text{ini}[sID], \text{res}[sID]) = (i^*, r^*) \wedge I[sID] = I[sID^*] \wedge R[sID] = R[sID^*]$	//matching sessions
$\wedge \text{type}[sID] \neq \text{type}[sID^*]\}$	
02 $\mathfrak{P}(sID^*) := \{sID \mid (\text{ini}[sID], \text{res}[sID]) = (i^*, r^*) \wedge I[sID] = I[sID^*] \wedge \text{type}[sID] = \text{"In"}$	//part. match. sess.
$\wedge \text{type}[sID] \neq \text{type}[sID^*]\}$	
03 AttackTable := Table 1	//by default we define PFS
04 if $X = \text{WFS}$ then AttackTable := Table 2	//if defining WFS use other table
05 for attack \in AttackTable:	
06 if attack = true :	
07 return true	
08 return false	

Fig. 5. UVALID_X verifies the validity of attacks against the UAKE protocol.

Definition 5. We define the game OW-UAKE- G_X for $X \in \{\text{PFS}, \text{WFS}\}$ as in Fig. 4. The advantage of an adversary \mathcal{A} against UAKE in this game is defined as

$$\text{Adv}_{\text{UAKE}}^X(\mathcal{A}) := \Pr[\text{OW-UAKE-}G_X^A \Rightarrow 1].$$

3.2 From WFS to PFS Secure UAKE

We construct a UAKE protocol UAKE_{PFS} with perfect forward secrecy from a UAKE protocol UAKE_{WFS} with weak forward secrecy and two hash functions H, H_{ukey} . The idea is that the session key of UAKE_{WFS} will be used twice: to derive the UAKE_{PFS} session key and to compute a key confirmation hash which is sent together with M_2 . An illustration of the protocol is given in Fig. 6.

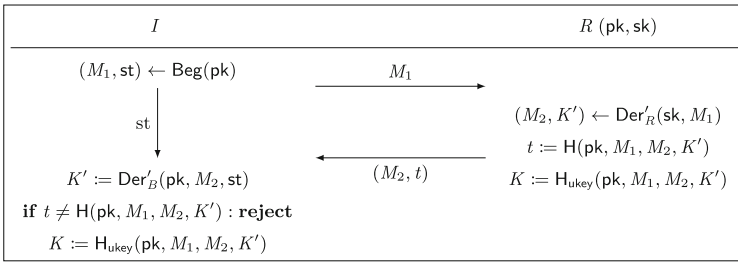


Fig. 6. Protocol $\text{UAKE}_{\text{PFS}} = (\text{Setup}, \text{KeyGen}, \text{Beg}, \text{Der}'_R, \text{Der}'_B)$ constructed from $\text{UAKE}_{\text{WFS}} = (\text{Setup}, \text{KeyGen}, \text{Beg}, \text{Der}_R, \text{Der}_B)$ and random oracles H, H_{ukey} .

Table 1. Attack table for UAKE describing valid attacks for perfect forward secrecy. An attack is regarded as an AND conjunction of variables with specified values as shown in the each line, where “-” means that this variable can take arbitrary value, “F” means “false”, “n/a” indicates that there is no state which can be revealed as no (partially) matching session exists.

		corr[r^*]	peerCorr[sID*]	type[sID*]	revState[sID*]	$\exists \text{sID} \in \mathfrak{M}(\text{sID}^*) :$ revState[sID]	$\mathfrak{M}(\text{sID}^*)$	$\exists \text{sID} \in \mathfrak{R}(\text{sID}^*) :$ revState[sID]	$\mathfrak{R}(\text{sID}^*)$
\mathcal{A} gets (i^*, r^*)									
(0) multiple partially matching sessions	-	-	-	-	-	-	-	-	> 1
(1) (-, long-term)	-	-	-	F	F	1	-	-	-
(2) (-, long-term)	-	-	“Re”	F	n/a	0	F	1	-
(3) (-, long-term)	-	F	“In”	F	n/a	0	n/a	0	-

Table 2. Attack table for UAKE describing valid attacks for weak forward secrecy. An attack is regarded as an AND conjunction of variables with specified values as shown in the each line, where “-” means that this variable can take arbitrary value, “**F**” means “false”, “n/a” indicates that there is no state which can be revealed as no (partially) matching session exists.

\mathcal{A} gets (i^*, r^*)	$\text{corr}[r^*]$	$\text{peerCorr}[sID^*]$	$\text{type}[sID^*]$	$\text{revState}[sID^*]$	$\exists sID \in \mathfrak{M}(sID^*) :$ $\text{revState}[sID]$	$ \mathfrak{M}(sID^*) $	$\exists sID \in \mathfrak{R}(sID^*) :$ $\text{revState}[sID]$	$ \mathfrak{R}(sID^*) $
(0) multiple partially matching sessions	-	-	-	-	-	-	-	> 1
(1) (-, long-term)	-	-	-	F	F	1	-	-
(2) (-, long-term)	-	-	“Re”	F	n/a	0	F	1
(3) (-, long-term)	F	-	“In”	F	n/a	0	n/a	0

Observe that the construction does not introduce any new primitives at all. Hence, correctness is preserved from the underlying UAKE_{WFS} .

Lemma 1. *If UAKE_{WFS} has correctness $\rho = 1 - 1/2^v$ for some $v \in \Omega(\kappa)$ then UAKE_{PFS} has overwhelming correctness at least ρ .*

Theorem 1 (UAKE_{WFS} to UAKE_{PFS}). *Let UAKE_{WFS} be $(1 - 1/2^v)$ -correct and with min-entropy μ . Let ζ be the lower bound for the dimensions of the tag space and key space. For any adversary \mathcal{A} against OW-UAKE-G_{PFS} with protocol UAKE_{PFS} , there exists an adversary \mathcal{B} against OW-UAKE-G_{WFS} with protocol UAKE_{WFS} such that*

$$\text{Adv}_{\text{UAKE}_{\text{PFS}}}^{\text{PFS}}(\mathcal{A}) \leq \text{Adv}_{\text{UAKE}_{\text{WFS}}}^{\text{WFS}}(\mathcal{B}) + \frac{N^2 + S^2 + Sq_{\text{RO}}}{2\mu} + \frac{S + q_{\text{RO}}^2 + q_{\text{Ch}}}{2\zeta} + \frac{2S}{2^v},$$

where N is the number of queries that \mathcal{A} and \mathcal{B} make to the key generation oracle, S is the number of sessions that \mathcal{A} and \mathcal{B} create, and q_{RO} and q_{Ch} are the number of random oracle and check queries that \mathcal{A} makes. Further, the running time of \mathcal{B} is about that of \mathcal{A} .

We give the full proof in the full version and we want to give a brief intuition here. It is indeed very similar to the proof for AKE in [26], adapted to the UAKE case. For this, note that the only difference between the weak and perfect forward security is that in the latter the adversary \mathcal{A} is allowed to query corrupt after the session is completed even if there is no matching session. We will show that due to the key confirmation tag, \mathcal{A} can never complete a session for which the peer was not corrupted. More specifically, \mathcal{A} has to forge a valid tag t . For this, it has to compute the underlying UAKE key and query it to the random oracle. Hence, we can construct a reduction which extracts the key and wins game UAKE_{WFS} .

4 Group Authenticated Key Exchange

We define group authenticated key exchange (GAKE), which is an N -party protocol, with $N > 2$, where all parties authenticate to each other. We consider three-round broadcast protocols where each round corresponds to a message broadcast. It is exemplified in Fig. 7.

We indicate with $\mathbf{P} = \{P_1, \dots, P_N\}$ the set of all potential members. Before the first run of the protocol, each party $P_n \in \mathbf{P}$ runs the algorithm `KeyGen` to get their own long-term public and secret keys $(\text{pk}_n, \text{sk}_n)$.

Our GAKE protocol allows all parties in a group $\mathbf{P}' \subseteq \mathbf{P}$ to establish a common secret group key. For a party P_n , we define $\mathcal{P}_n := \mathbf{P}' \setminus \{P_n\}$ the set of the peers from the point of view of P_n . The following provides a detailed explanation of how our GAKE protocol works and offers proper syntax.

SYNTAX. A group authenticated key exchange protocol $\text{GAKE} := (\text{Setup}, \text{KeyGen}, \text{Begin}, \text{Respond}, \text{Final}, \text{Derive})$ consists of six polynomial-time algorithms:

- $\text{par} \leftarrow \text{Setup}(1^\kappa)$: The probabilistic setup algorithm `Setup` takes as input the security parameter κ in unary and returns global system parameters par that implicitly define message space \mathcal{T} , the public key space \mathcal{PK} , the secret key space \mathcal{SK} and the key space \mathcal{K} .
- $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{par})$: The probabilistic key generation algorithm `KeyGen` takes as input the parameters par and returns a public key $\text{pk} \in \mathcal{PK}$ and a secret key $\text{sk} \in \mathcal{SK}$.
- $(m, \text{st}) \leftarrow \text{Begin}(\text{sk}, \{\text{pk}_j\}_{j \in \mathcal{P}})$: The probabilistic first round algorithm `Begin` takes as input a secret key sk and a set of public keys $\{\text{pk}_j\}_{j \in \mathcal{P}}$ of the peers $\mathcal{P} \subset \mathbb{N}$ and returns a message $m \in \mathcal{T}$ and a state st .
- $(\hat{m}, \text{st}) \leftarrow \text{Respond}(\text{sk}, \text{st}, \mathcal{M})$: The probabilistic second round algorithm `Respond` takes a secret key sk , a state st , and a set \mathcal{M} of extended messages $\mathcal{M} = \{(i, m)\}$. Each extended message is a pair consisting of an index $i \in \mathbb{N}$ and a message $m \in \mathcal{T}$. The algorithm returns a message $\hat{m} \in \mathcal{T}$ and an updated state st .
- $(\bar{m}, \text{st}) \leftarrow \text{Final}(\text{sk}, \text{st}, \hat{\mathcal{M}})$: The probabilistic third round algorithm `Final` takes a secret key sk , a state st , and a set of extended messages $\hat{\mathcal{M}} = \{(i, \hat{m})\}$ with index $i \in \mathbb{N}$ and message $\hat{m} \in \mathcal{T}$, and returns a message $\bar{m} \in \mathcal{T}$ and an updated state st .
- $K := \text{Derive}(\text{sk}, \text{st}, \bar{\mathcal{M}})$: The deterministic derivation algorithm `Derive` takes a secret key sk , a state st and a set of extended messages $\bar{\mathcal{M}} = \{(i, \bar{m})\}$ with index $i \in \mathbb{N}$ and $\bar{m} \in \mathcal{T}$, and returns a group key $K \in \mathcal{K}$.

Definition 6 (Correctness GAKE). *Given N parties, we say that GAKE is ρ -correct, if for any $\text{par} \leftarrow \text{Setup}(1^\kappa)$ we have:*

$$\Pr \left[K_1 = \dots = K_N \mid \begin{array}{l} \forall i \in [N] : (\text{pk}_i, \text{sk}_i) \leftarrow \text{KeyGen}(\text{par}), \\ (m_i, \text{st}_i) \leftarrow \text{Begin}(\text{sk}_i, \{\text{pk}_j\}_{j \in [N] \setminus \{i\}}), \\ (\hat{m}_i, \text{st}_i) \leftarrow \text{Respond}(\text{sk}_i, \text{st}_i, \mathcal{M}_i), \\ (\bar{m}_i, \text{st}_i) \leftarrow \text{Final}(\text{sk}_i, \text{st}_i, \hat{\mathcal{M}}_i), \\ K_i := \text{Derive}(\text{sk}_i, \text{st}_i, \bar{\mathcal{M}}_i) \end{array} \right] \geq \rho,$$

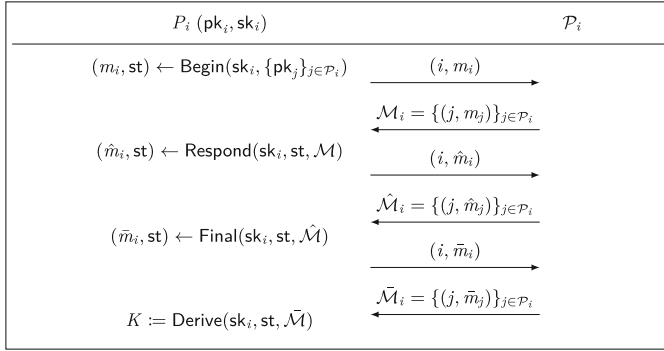


Fig. 7. Execution of the GAKE algorithms in a protocol run.

where $\mathcal{M}_i = \{(j, m_j)\}_{j \in [N] \setminus \{i\}}$, $\hat{\mathcal{M}}_i = \{(j, \hat{m}_j)\}_{j \in [N] \setminus \{i\}}$, and $\bar{\mathcal{M}}_i = \{(j, \bar{m}_j)\}_{j \in [N] \setminus \{i\}}$ and the probability is over the random coins consumed by the algorithms of GAKE.

To each new run of the group key protocol we assign a unique identification number sID and variables which are defined relative to sID . We call such a run a session.

- $\text{holder}[\text{sID}] \in [N]$ denotes the holder of the session.
- $\text{peers}[\text{sID}] \subseteq [N]$ denotes the peers of the session.
- $I[\text{sID}], R[\text{sID}], F[\text{sID}]$ denotes the extended message sent by the holder in the first, second and third round respectively.
- $\mathcal{M}[\text{sID}], \hat{\mathcal{M}}[\text{sID}], \bar{\mathcal{M}}[\text{sID}]$ denotes the extended messages received by the holder in the first, second and third round respectively.
- $\text{state}[\text{sID}]$ denotes the state.
- $\text{gKey}[\text{sID}]$ denotes the group key.
- $\text{stage}[\text{sID}] \in \{2, 3, 4, 5\}$ is used to model that the algorithms of each session are executed in a specific order.

Moreover, we use the following boolean values to store which queries the adversary made.

- $\text{corr}[n]$ denotes if long-term secret key of party P_n has been given to the adversary.
- $\text{revealed}[\text{sID}]$ denotes if the group key has been given to the adversary.
- $\text{revState}[\text{sID}]$ denotes if the state has been given to the adversary.
- $\text{peersCorr}[\text{sID}]$ denotes if one of the peers is corrupted and its long-term key has already been given to the adversary at the time the group key is derived.

Let us now define what it means for two sessions to be partially matching for GAKE protocols. This follows the same motivation as in the definition for the two-party case.

Definition 7 (Partially Matching Session). *Two sessions sID and sID^* are partially matching if the following conditions are satisfied.*

1. *The sessions have distinct holders, $\text{holder}[sID] \neq \text{holder}[sID^*]$.*
2. *The extended messages in the first round are identical, $I[sID] \cup \mathcal{M}[sID] = I[sID^*] \cup \mathcal{M}[sID^*]$*
3. *The extended messages in the second round are identical, $R[sID] \cup \hat{\mathcal{M}}[sID] = R[sID^*] \cup \hat{\mathcal{M}}[sID^*]$*

Definition 8 (Matching Session). *Two sessions sID and sID^* are matching if they are partially matching and additionally, we have:*

4. *The extended messages in the third round are identical, $F[sID] \cup \bar{\mathcal{M}}[sID] = F[sID^*] \cup \bar{\mathcal{M}}[sID^*]$*

SECURITY NOTION. We give the full description of the security game in Fig. 8. In contrast to the UAKE game, this game models key indistinguishability. The interfaces are however very similar. We allow the adversary to create parties via KEYGENERATION. It can create groups and send messages to its members via oracles SESSION_B, SESSION_R, SESSION_F, DER. Here, we not only allow to reveal the state and long-term keys of parties, but also session keys. Security is captured by the TEST oracle which can be queried multiple times. All queries are answered with the same bit b .

Similar to UAKE, we use an attack table to describe valid attacks. Here, we only define the stronger notion of perfect forward secrecy in Table 3 since this is our target notion. One can define weak forward secrecy by modifying the table similar to the UAKE notions. Intuitively, we aim for the strongest notion possible where the adversary is allowed to query *either* the long-term key *or* the secret state of any party in the group, even if the corresponding session of any group member will later be queried to TEST.

We now describe Table 3 in more detail. As for UAKE, we let the adversary win directly if the protocol does not have sufficient entropy. Further, we iterate over all peers of a session to detect trivial attacks. More specifically, we look at the holder of the session and then at each group member individually. Depending on whether this group member has the same view as the holder, which we determine by the checking whether they have a (partially) matching session, we allow the adversary to reveal long-term keys or states.

- Attacks (1)–(4) deal with matching sessions, where we essentially capture all combinations of reveal queries.
- Attacks (5)–(8) capture partially matching sessions which are the same as (1)–(4), except that we need to look at the state of those sessions in set \mathfrak{P} .
- Attacks (9)–(10) look at sessions where the peer does not hold a session with the same view (hence, the adversary actively modified communication). Here, we need to be more restrictive since the adversary can pick some of the states itself, in which case we cannot allow it to also reveal the long-term key.

Definition 9. We define the game $\text{GAKE-G}_{\text{PFS},b}$ for $b \in \{0, 1\}$ as in Fig. 8. The advantage of an adversary \mathcal{A} against GAKE in this game is defined as

$$\text{Adv}_{\text{GAKE}}^{\text{PFS}}(\mathcal{A}) := \left| \Pr[\text{GAKE-G}_{\text{PFS},1}^{\mathcal{A}} \Rightarrow 1] - \Pr[\text{GAKE-G}_{\text{PFS},0}^{\mathcal{A}} \Rightarrow 1] \right|.$$

<u>GAMES</u> GAKE-G_b	<u>SESSION_F</u> (sID, \mathcal{M})
<u>INITIALIZE</u>	
00 cnt ₁ := 0 //session counter	43 parse $\{(j, \hat{m}_j)\}_{j \in \mathcal{P}_i} := \hat{\mathcal{M}}$
01 cnt ₂ := 0	44 if stage[sID] $\neq 3$: return \perp
02 $N := 0$	45 if $ \mathcal{M} \neq \text{peers[sID]} $: return \perp
03 $\mathcal{S} := \emptyset$ //set of test sessions	46 if $\{j \mid \exists \hat{m}_j \text{ s.t. } (j, \hat{m}_j) \in \hat{\mathcal{M}}\} \neq \text{peers[sID]}_1$: return \perp
04 par $\leftarrow \text{Setup}(1^\kappa)$	47 $(i, \mathcal{P}_i) := (\text{holder[sID]}_1, \text{peers[sID]}_1)$
05 return par	48 $\text{peersCorr[sID]} := \bigvee_{j \in \mathcal{P}_i} \text{corr}[j]$
<u>KEYGENERATION</u> (par)	49 $\mathcal{M}[\text{sID}] := \mathcal{M}$
06 $N++$	50 $(\bar{m}_i, \text{st}) \leftarrow \text{Final}(\text{sk}_i, \text{state[sID]}, \mathcal{M})$
07 $(\text{pk}_N, \text{sk}_N) \leftarrow \text{KeyGen}(\text{par})$	51 $F[\text{sID}] := \{(i, \bar{m}_i)\}$ //store third extended message
08 return pk_N	52 $\text{state[sID]} := \text{st}$
	53 $\text{stage[sID]} := 4$
	54 return (sID, \bar{m}_i)
<u>FINALIZE</u> (b')	<u>DER</u> (sID, $\bar{\mathcal{M}}$)
09 for sID* $\in \mathcal{S}$: //multiple test sessions	55 parse $\{(j, \bar{m}_j)\}_{j \in \mathcal{P}_i} := \bar{\mathcal{M}}$
10 for $P \in \text{peers[sID]}^*$:	56 if stage[sID] $\neq 4$: return \perp
11 if $\text{GVALID}_{\text{PFS}}(\text{sID}^*, P) = \text{false}$:	57 if $ \bar{\mathcal{M}} \neq \text{peers[sID]} $: return \perp
12 return 0 //no valid attack	58 if $\{j \mid \exists \bar{m}_j \text{ s.t. } (j, \bar{m}_j) \in \bar{\mathcal{M}}\} \neq \text{peers[sID]}$: return \perp
13 return b'	59 if $\text{gKey[sID]} \neq \perp$: return \perp //key already computed
<u>SESSION_B</u> (i, \mathcal{P}_i)	60 $(i, \mathcal{P}_i) := (\text{holder[sID]}_1, \text{peers[sID]}_1)$
14 if $\mathcal{P}_i \not\subseteq [N]$: return \perp //subset of known pks	61 $\text{peersCorr[sID]} := \bigvee_{j \in \mathcal{P}_i} \text{corr}[j]$
15 if $i \notin [N]$: return \perp //key material exists	62 $\bar{\mathcal{M}}[\text{sID}] := \bar{\mathcal{M}}$
16 if $i \in \mathcal{P}_i$: return \perp //all public keys distinct	63 $K := \text{Derive}(\text{sk}_i, \text{state[sID]}, \bar{\mathcal{M}})$
17 cnt ₁ ++	64 $\text{gKey[sID]} := K$
18 sID ₁ := cnt ₁	65 $\text{stage[sID]} := 5$
19 sID ₂ := \perp	66 return ε
20 sID := (sID ₁ , sID ₂)	<u>REVEAL</u> (sID)
21 $\text{holder[sID]}_1 := i$	67 revealed[sID]} := \text{true}
22 $\text{peers[sID]}_1 := \mathcal{P}_i$	68 return gKey[sID]
23 $(m_i, \text{st}) \leftarrow \text{Begin}(\text{sk}_i, \{\text{pk}_j\}_{j \in \mathcal{P}_i})$	<u>REV-STATE</u> (sID)
24 $I[\text{sID}] := \{(i, m_i)\}$ //store first extended message	69 $\text{revState[sID]} := \text{true}$
25 $\text{state[sID]} := \text{st}$	70 return state[sID]
26 $\text{stage[sID]} := 2$	<u>CORRUPT</u> (n)
27 return (sID, m_i)	71 $\text{corr}[n] := \text{true}$
<u>SESSION_R</u> (sID, \mathcal{M})	72 return sk_n
28 parse $\{(j, m_j)\}_{j \in \mathcal{P}_i} := \mathcal{M}$	<u>TEST</u> (sID)
29 if stage[sID] $\neq 2$: return \perp //session not created	73 if sID $\in \mathcal{S}$: return \perp
30 if $ \mathcal{M} \neq \text{peers[sID]} $: return \perp //correct no. msgs.	74 if $\text{gKey[sID]} = \perp$: return \perp
31 if $\{j \mid \exists m_j \text{ s.t. } (j, m_j) \in \mathcal{M}\} \neq \text{peers[sID]}$: return \perp //all partners have sent messages	75 $\mathcal{S} := \mathcal{S} \cup \{\text{sID}\}$
32 $(i, \mathcal{P}_i) := (\text{holder[sID]}_1, \text{peers[sID]}_1)$	76 $K_0^* := \text{gKey[sID]}$
33 cnt ₂ ++	77 $K_1^* \leftarrow K$
34 sID ₂ := cnt ₂	78 return K_0^*
35 sID := (sID ₁ , sID ₂)	
36 $\text{peersCorr[sID]} := \bigvee_{j \in \mathcal{P}_i} \text{corr}[j]$	
37 $\mathcal{M}[\text{sID}] := \mathcal{M}$	
38 $(\hat{m}_i, \text{st}) \leftarrow \text{Respond}(\text{sk}_i, \text{state[sID]}, \mathcal{M})$	
39 $R[\text{sID}] := \{(i, \hat{m}_i)\}$ //store second extended message	
40 $\text{state[sID]} := \text{st}$	
41 $\text{stage[sID]} := 3$	
42 return (sID, \hat{m}_i)	

Fig. 8. Games $\text{GAKE-G}_{\text{PFS},b}$ for GAKE , where $b \in \{0, 1\}$. \mathcal{A} has access to oracles $\mathcal{O} := \{\text{SESSION}_B, \text{SESSION}_R, \text{SESSION}_F, \text{DER}, \text{REVEAL}, \text{REV-STATE}, \text{CORRUPT}, \text{TEST}\}$. Helper procedure $\text{GVALID}_{\text{PFS}}$ is defined in Fig. 9. If there exists any test session which is not valid, the game will return 0.

GVALID _{PFS} (sID*, P)	
00	$i^* := \text{holder}[sID^*]$
01	$\mathfrak{M}(sID^*, P) := \{sID \mid \text{holder}[sID] = P \wedge I[sID] \cup \mathcal{M}[sID] = I[sID^*] \cup \mathcal{M}[sID^*]$ $\quad \wedge R[sID] \cup \tilde{\mathcal{M}}[sID] = R[sID^*] \cup \tilde{\mathcal{M}}[sID^*] \wedge F[sID] \cup \tilde{\mathcal{M}}[sID] = F[sID^*] \cup \tilde{\mathcal{M}}[sID^*]\}$
02	$\mathfrak{P}(sID^*, P) := \{sID \mid \text{holder}[sID] = P \wedge I[sID] \cup \mathcal{M}[sID] = I[sID^*] \cup \mathcal{M}[sID^*] \text{ //part. match. sess. to } P$ $\quad \wedge R[sID] \cup \tilde{\mathcal{M}}[sID] = R[sID^*] \cup \tilde{\mathcal{M}}[sID^*]\}$
03	if revealed[sID*] or $(\exists sID \in \mathfrak{M}(sID^*, P) : \text{revealed}[sID] = \mathbf{true})$ //session or partner revealed or $\exists sID \in \mathfrak{M}(sID^*, P)$ s. t. sID $\in \mathcal{S}$: //Test was asked for two partnered sessions
04	return false
05	for attack \in Table 3
06	if attack = true :
07	return true
08	return false

Fig. 9. GVALID_{PFS} captures perfect forward secrecy and verifies the validity of attacks against the GAKE protocol.

5 GAKE from UAKE

We construct a GAKE protocol from a UAKE protocol as shown in Fig. 10. Each party will run the UAKE protocol twice to generate two fresh symmetric keys. While doing this, each party uses the first UAKE run to authenticate itself to the predecessor (acting as responder in the UAKE), and the other to authenticate its successor (acting as initiator). In the second phase, each party will encrypt the key that it shares with its predecessor to its successor. In this way, shared keys will only be made available to parties that have been authenticated. In the final step, each party will step-wisely compute all the pairwise shared keys and use them to derive the final group key.

To protect critical information from the attacker, we will encrypt the state information with the long-term key. The state information consists of all the information that needs to be pertained between rounds for the the protocol to work properly. To simplify, we will in our description encrypt all state information. Indeed some of the state information of sessions can be derived by the attacker publicly. Encrypting the entire state information allows us to abstract and more generically describe the mechanisms we use.

We will rely on perfect forward secrecy of UAKE to achieve perfect forward secrecy of GAKE, hence we refer to the protocols as UAKE_{PFS} and GAKE_{PFS}, respectively.

5.1 Correctness

It can be shown that the final construction has overwhelming correctness if the underlying UAKE has overwhelming correctness.

Lemma 2. *Consider the construction in Fig. 10. If UAKE_{PFS} has overwhelming correctness $\rho = 1 - 1/2^v$ for some $v \in \Omega(\kappa)$ and the attacker makes q queries overall, then GAKE_{PFS} has overwhelming correctness at least $1 - q/2^v$.*

Proof. Assume UAKE_{PFS} has overwhelming correctness $\rho = 1 - 1/2^v$ for some $v \in \Omega(\kappa)$. First, observe that by setup this is the only source of non-perfect

Table 3. Attack table describing valid attacks for full perfect forward secrecy (PFS). An attack is regarded as an AND conjunction of variables with specified values as shown in the each line, where “-” means that this variable can take arbitrary value, “**F**” means “false”, “n/a” indicates that the result is trivially “false” because of the definition of (partially) matching sessions.

\mathcal{A} gets (holder, P)	$\text{corr}[i^*]$	$\text{corr}[P]$	$\text{peersCorr}[\text{sID}^*]$	$\text{revState}[\text{sID}^*]$	$\exists \text{sID} \in \mathfrak{M}(\text{sID}^*, P) :$ $\text{revState}[\text{sID}]$	$\exists \mathfrak{M}(\text{sID}^*, P)$	$\exists \text{sID} \in \mathfrak{R}(\text{sID}^*, P) :$ $\text{revState}[\text{sID}]$	$\exists \mathfrak{R}(\text{sID}^*, P)$
(0) multiple partially matching sessions	-	-	-	-	-	-	-	> 1
(1) (long-term, long-term)	-	-	-	F	F	1	-	-
(2) (state, state)	F	F	-	-	-	1	-	-
(3) (long-term, state)	-	F	-	F	-	1	-	-
(4) (state, long-term)	F	-	-	-	F	1	-	-
(5) (long-term, long-term)	-	-	F	F	n/a	0	F	1
(6) (state, state)	F	F	-	-	n/a	0	-	1
(7) (long-term, state)	-	F	-	F	n/a	0	-	1
(8) (state, long-term)	F	-	F	-	n/a	0	F	1
(9) (long-term, long-term)	-	-	F	F	n/a	0	n/a	0
(10) (state, state)	F	F	-	-	n/a	0	n/a	0

correctness in the entire protocol construction. Also observe that if UAKE_{PFS} has no correctness errors at all, then we will not have any correctness error in our GAKE_{PFS} construction as well. So we only have to analyse the influence of UAKE_{PFS} on the overall correctness. Now, since a single application of UAKE_{PFS} has overwhelmingly high correctness ρ , a q -time application of UAKE_{PFS} will result in a correctness of at least $(1 - 1/2^v)^q$. This can be lower bounded via $(1 - 1/2^v)^q \geq 1 - q/2^v$ for some arbitrary polynomial $q = q(\kappa)$ due the Bernoulli’s inequality, which shows that the resulting correctness is still overwhelming. \square

5.2 Security

We now prove the security of our construction. Informally, if UAKE has perfect forward secrecy, then the resulting GAKE also has perfect forward secrecy. This is captured in the following theorem.

Theorem 2 (UAKE to GAKE). *For any adversary \mathcal{A} against $\text{GAKE-G}_{\text{PFS},b}$ with protocol GAKE_{PFS} with N parties that establish at most S sessions and*

<pre> Setup(1^κ) 00 par \leftarrow Setup_{UAKE}(1^κ) 01 return par KeyGen(par) 02 (pk_{UAKE}, sk_{UAKE}) \leftarrow KeyGen_{UAKE}(par) 03 $k \leftarrow \{0, 1\}^\kappa$ 04 return (pk, sk) := (pk_{UAKE}, (sk_{UAKE}, k)) Begin(sk_{i}, pk_{i}, {pk_{j}}_{$j \in \mathcal{P}_i$}) 05 compute the bijection $\pi(\cdot)$ with $\pi(\{ \mathcal{P}_i + 1\}) = \mathcal{P}_i \cup \{i\}$ s.t. PK' := (pk'₁, ..., pk'_{$\mathcal{P}_i +1$}) := (pk_{$\pi(1)$}, ..., pk_{$\pi(\mathcal{P}_i +1)$}) is lexicographically ordered 06 $\hat{h} := H_k(\text{PK})$ 07 ($M_{1, \text{prd}[i, \pi]}$, st_{UAKE, prd[i, π]}) \leftarrow Beg(pk_{prd[i, π]}) 08 $m_i := M_{1, \text{prd}[i, \pi]}$ 09 st_{i} := (pk_{prd[i, π]}, m_i, h, prd[i, π], scc[i, π], st_{UAKE, prd[i, π]}) 10 $IV \leftarrow \{0, 1\}^\kappa$ 11 $w := H_\pi(IV, k) \oplus \text{st}_i$ 12 st'_{i} := (IV, w) 13 return (m_i, st'_{i}) Respond(sk_{i}, st'_{i}, \mathcal{M}_i) 14 parse {(j, m_j)}_{$j \in \mathcal{P}_i$} := \mathcal{M}_i 15 parse (IV, w) := st'_{i} 16 st_{i} := H_{π}(IV, k) \oplus w 17 parse (pk_{prd[i, π]}, m_i, h, prd[i, π], scc[i, π], st_{UAKE, prd[i, π]}) := st_{i} 18 $\mathcal{M}'_i := ((1, m_1), \dots, (\mathcal{P}_i + 1, m'_{ \mathcal{P}_i +1}))$:= (($\pi(1)$, $m_{\pi(1)}$), ..., ($\pi(\mathcal{P}_i + 1)$, $m_{\pi(\mathcal{P}_i +1)}$)) 19 $\hat{h} := H_{\text{ext}}(\mathcal{M}'_i, h)$ 20 ($M_{2, i}$, K_{UAKE, i}) \leftarrow Der_R(sk_{UAKE, i}, m_{scc[i, π]}) 21 $\tilde{m}_i := M_{2, i}$ 22 K_{UAKE, i} := H_{prkey}(K_{UAKE, i}, \tilde{m}_i, \hat{h}) 23 st_{i} := (pk_{prd[i, π]}, K_{UAKE, i}, \tilde{m}_i, \hat{h}, prd[i, π], scc[i, π], st_{UAKE, prd[i, π]}) 24 $\tilde{IV} \leftarrow \{0, 1\}^\kappa$ 25 $\tilde{w} := H_\pi(\tilde{IV}, k) \oplus \text{st}_i$ 26 st'_{i} := (\tilde{IV}, \tilde{w}) 27 return ($\tilde{m}_i, \text{st}'_i$) </pre>	<pre> Final(sk_{i}, st'_{i}, $\tilde{\mathcal{M}}_i$) 28 parse {(j, \tilde{m}_j)}_{$j \in \mathcal{P}_i$} := $\tilde{\mathcal{M}}_i$ 29 parse (IV, \tilde{w}) := st'_{i} 30 st_{i} := H_{π}(IV, k) \oplus \tilde{w} 31 parse (pk_{prd[i, π]}, K_{UAKE, i}, \tilde{m}_i, \hat{h}, prd[i, π], scc[i, π], st_{UAKE, prd[i, π]}) := st_{i} 32 $\mathcal{M}'_i := ((1, \tilde{m}'_1), \dots, (\mathcal{P}_i + 1, \tilde{m}'_{ \mathcal{P}_i +1}))$:= (($\pi(1)$, $\tilde{m}_{\pi(1)}$), ..., ($\pi(\mathcal{P}_i + 1)$, $\tilde{m}_{\pi(\mathcal{P}_i +1)}$)) 33 K_{UAKE, prd[i, π]} := Der_B(pk_{prd[i, π]}, $\tilde{m}_{\text{prd}[i, \pi]}$, st_{UAKE, prd[i, π]}) 34 K_{UAKE, prd[i, π]} := H_{prkey}(K_{UAKE, prd[i, π]}, $\tilde{m}_{\text{prd}[i, \pi]}$, \hat{h}) 35 K_{$i, \text{scc}[i, \pi]$} := H_{ukey}(K_{UAKE, i}, \mathcal{M}'_i) 36 K_{prd[i, π]} := H_{ukey}(K_{UAKE, prd[i, π]}, $\tilde{\mathcal{M}}'_i$) 37 <math>\tilde{m}_i := H_{\text{tag}}(K_{\text{prd}[i, \pi]}, \tilde{m}_i) \oplus K_{$i, \text{scc}[i, \pi]$} 38 st_{i} := (K_{prd[i, π]}, K_{$i, \text{scc}[i, \pi]$}, $\tilde{m}_i, \text{prd}[i, \pi]$, scc[i, π]) 39 $\tilde{IV} \leftarrow \{0, 1\}^\kappa$ 40 $\tilde{w} := H_\pi(\tilde{IV}, k) \oplus \text{st}_i$ 41 st'_{i} := (\tilde{IV}, \tilde{w}) 42 return ($\tilde{m}_i, \text{st}'_i$) Derive(sk_{$i$}, st'_{$i$}, $\tilde{\mathcal{M}}_i$) 43 parse {(j, \tilde{m}_j)}_{$j \in \mathcal{P}_i$} := $\tilde{\mathcal{M}}_i$ 44 parse (IV, \tilde{w}) := st'_{i} 45 st_{i} := H_{π}(IV, k) \oplus \tilde{w} 46 parse (K_{prd[i, π]}, K_{$i, \text{scc}[i, \pi]$}, $\tilde{m}_i, \text{prd}[i, \pi]$, scc[i, π]) := st_{i} 47 $j := \text{scc}[i, \pi]$ 48 repeat 49 K_{$j, \text{scc}[i, \pi]$} := H_{tag}(K_{prd[i, π]}, j) \oplus \tilde{m}_j 50 $j := \text{scc}[i, \pi]$ 51 until $j = i$ 52 $\tilde{\mathcal{M}}'_i := ((1, \tilde{m}'_1), \dots, (\mathcal{P}_i + 1, \tilde{m}'_{ \mathcal{P}_i +1}))$:= (($\pi(1)$, $\tilde{m}_{\pi(1)}$), ..., ($\pi(\mathcal{P}_i + 1)$, $\tilde{m}_{\pi(\mathcal{P}_i +1)}$)) 53 $\tilde{K} := (K_{\pi(1)}, \dots, K_{\pi(\mathcal{P} +1)})$ 54 K := H_{gkey}($\tilde{K}, \tilde{\mathcal{M}}'_i$) 55 return K </math></pre>
---	--

Fig. 10. Generic construction of GAKE_{PFS} from UAKE_{PFS} = (Setup_{UAKE}, KeyGen_{UAKE}, Beg, Der_R, Der_B).

issues at most q queries to the oracles, there exists an adversary \mathcal{B} against OW-UAKE-G_{PFS} of a protocol UAKE_{PFS} with min-entropy μ and correctness $(1 - 1/2^v)$, such that

$$\begin{aligned} \text{Adv}_{\text{GAKE}_{\text{PFS}}}^{\text{PFS}}(\mathcal{A}) &\leq \text{Adv}_{\text{UAKE}_{\text{PFS}}}^{\text{PFS}}(\mathcal{B}) + \frac{Sq_{RS} + Nq_C}{2^\kappa} + \frac{Sq_{H_{\text{ukey}}}}{2^\tau} + \frac{Sq_{H_{\text{tag}}}}{2^\tau} \\ &\quad + \frac{Sq_{H_{\text{gkey}}}}{|\mathcal{K}_{\text{GAKE}_{\text{PFS}}}|} + \frac{N^2 + S^2}{2^\mu} + \frac{N^2 + 3S^2}{2^\kappa} + \frac{q_{RO}^2}{2^\zeta} + \frac{q}{2^v}, \end{aligned}$$

where all hash functions are modeled as random oracles and the running time of \mathcal{B} is about that of \mathcal{A} .

Let us first sketch the proof. We proceed in a series of games. In the first steps, we make sure that all outputs of the sessions are distinct by relying on the min-entropy of the UAKE. Next, we exclude collisions in the random oracles. Finally, we assume that all UAKE runs feature correctness. Each time this accounts only for a statistically small change in the success probability of the attacker. Next we make the state of each stage independent of all the initial values and secret parameters. At the same time, we ensure consistency by adapting

the random oracle used for state encryption to generate an output - on the fly - that is used to encrypt the state. In fact, the reduction creates it such that it can successfully appear to have encrypted the state beforehand. In this step, we essentially exploit that the hash function in the state encryption is modeled as a random oracle. After that, we change how the keys $K'_{\text{UAKE},i}$ are computed and make them independent of all previously computed values while guaranteeing consistency with all the queries of the attacker. Again, we exploit that the underlying hash function is used as a random oracle. We now have that uKey' is now independent of uKey . Now we partition the space of all sessions into two categories.

- 1) The set of sessions that still, when tested, could amount to a subset S of valid attacks (with respect to the predicate GVALID).
- 2) The set of sessions for which we already have certainty, that they can never amount to a valid attack in S .

Specifically, we will show that sessions of type 1) imply the existence of an algorithm that breaks the underlying UAKE scheme. Put differently, the attacks in 1) will correspond to an attacker that can compute the UAKE key in a non-trivial way. For the remaining sessions we will next apply an argument which guarantees that the keys $K'_{\text{UAKE},\text{prd}[\text{holder}[\text{sID}],\pi]}$ and $K'_{\text{UAKE},\text{holder}[\text{sID}]}$ are indistinguishable from random. This accounts for an additional statistically small change in the success probability. As a result, we now only consider group keys that are indistinguishable from random (with overwhelming probability).

Proof. Let \mathcal{A} be an adversary against $\text{GAKE-G}_{\text{PFS},b}$ with protocol GAKE_{PFS} as defined in Fig. 10. We consider the sequence of games described below and give their detailed pseudocode in the full version.

$\text{GAME } G_{0,b}$. This is the same as $\text{GAKE-G}_{\text{PFS},b}$, except for small changes. We store the un-encrypted state in an additional variable $\text{state}'[\text{sID}]$ and do not decrypt it explicitly. This is only conceptual. We also implicitly exclude collisions and if a collision happens at any time in the game, the experiment aborts. We also make sure that key pairs and messages are distinct. Using the fact that UAKE_{PFS} has min-entropy μ , the upper bound for key collisions is $N^2/2^\mu$ and for message collisions, it is $S^2/2^\mu$. Moreover, we assume that values k_n , for $n \in [N]$, and IV, \hat{IV}, \bar{IV} (at most S for each one) are distinct, and this is provided with probability at most $(N^2 + 3S^2)/2^\kappa$. In the end, we aim for all random oracle outputs to be unique. Assuming ζ is the lower bound for all dimensions of the random oracle outputs, collisions are excluded with a probability of at most $q_{RO}^2/2^\zeta$, where $q_{RO} \leq q$. All the probabilities above also follow from the birthday bound. Finally, in this step we abort if any of the UAKE runs of the challenger do not feature correctness. However, as analysed before this only happens with probability $q/2^v$.

We get

$$\begin{aligned} & \left| \Pr[\text{GAKE-G}_{\text{PFS},0}^A \Rightarrow 1] - \Pr[\text{GAKE-G}_{\text{PFS},1}^A \Rightarrow 1] \right| \leq \\ & \left| \Pr[G_{0,0}^A \Rightarrow 1] - \Pr[G_{0,1}^A \Rightarrow 1] \right| + \frac{N^2 + S^2}{2^\mu} + \frac{q_{KG}^2 + 3S^2}{2^\kappa} + \frac{q_{RO}^2}{2^\zeta} + \frac{q}{2^v}. \end{aligned}$$

GAME $G_{1,b}$. In games $G_{1,b}$, we make the state of each stage independent of all the initial values and secret parameters. At the same time, we ensure consistency by adapting the random oracle H_{st^*} to generate an output - on the fly - that is used to encrypt the state (actually the reduction creates it such that it can successfully appear to have encrypted the state beforehand). Now the initial values are computed in the REV-STATE oracle and the long-term secret values k_n are computed in the CORRUPT oracle. We raise flag BAD_{IV} and abort if the REV-STATE oracle chooses an initial value IV that was issued, together with the secret key of the corresponding holder of the session, to the H_{st^*} oracle before. The probability that BAD_{IV} is raised for one specific IV is at most $q_{RS}/2^\kappa$, where q_{RS} indicates the number of queries issued to the REV-STATE oracle and $q_{RS} \leq q$. An union bound gives us

$$\Pr[BAD_{IV}] \leq \frac{Sq_{RS}}{2^\kappa}. \quad (1)$$

We also raise flag BAD_k and abort if, for the chosen secret value k_n computed by the CORRUPT oracle, there exist an initial value IV such that both were issued to the H_{st^*} oracle before. The probability that BAD_k is raised for one specific k_n is at most $q_C/2^\kappa$, where q_C indicates the number of queries issued to the CORRUPT oracle and $q_C \leq q$. Again, an union bound gives us

$$\Pr[BAD_k] \leq \frac{Nq_C}{2^\kappa}. \quad (2)$$

Then, from Eqs. 1 and 2 we have

$$|\Pr[G_{1,b}^A \Rightarrow 1] - \Pr[G_{0,b}^A \Rightarrow 1]| \leq \frac{Sq_{RS} + Nq_C}{2^\kappa}.$$

GAME $G_{2,b}$. This is a bridging step. We essentially change how the keys $K'_{UAKE,i}$ are computed and make them independent of all previously computed values while guaranteeing consistency with all the queries of the attacker. We also introduce two helper variables, $uKey$ and $uKey'$, that store the keys $K_{UAKE,i}$ and $K'_{UAKE,i}$ for later use if needed.

$$\Pr[G_{2,b}^A \Rightarrow 1] = \Pr[G_{1,b}^A \Rightarrow 1].$$

GAME $G_{3,b}$. In the previous game we have already changed the way $uKey'$ is computed. In particular, it is now independent of $uKey$ while the simulation is still consistent. This holds for all sessions. Now we partition the space of all sessions into two categories.

- 1) The set of sessions that still, when tested, could amount to a subset S of valid attacks (with respect to the predicate G_{VALID}).

- 2) The set of sessions for which we already have certainty, that they can never amount to a valid attack in S .

Specifically, we will show that sessions of type 1) imply the existence of an algorithm that breaks the underlying UAE scheme. Intuitively, the attacks in 1) will correspond to an attacker that can compute the UAE key in a non-trivial way. Essentially, the conditions require sID to be a session that has successfully computed an UAE key in $SESSION_R$ (or $SESSION_F$) via a successive run of Beg and Der_B (or Der_R). We now need to show that whenever this happens, we can immediately break the security of the underlying UAE. So we set a flag BAD in case some preliminary conditions are fulfilled. This is the case if there is a specific query $w = (K, m, h)$ to the random oracle H_{prkey} such that:

- i) the query w has never been queried before to H_{prkey} ;
- ii) there exists a session sID that has helper variable $uKey$ – which is either $K_{UAE,holder[sID]}$ of the holder of that session (case (a)) or $K_{UAE,prd[holder[sID],\pi]}$ of the predecessor (case (b)) – be equal to K ;
- iii) the hash of the context so far $\hat{h}[sID] = H_{ctx}(\mathcal{M}_i[sID], h[sID])$ is equal to h ;
- iv) the message m is equal to the message $m_{holder[sID]}$ in case (a) and $m_{prd[holder[sID],\pi]}$ in case (b).

Now, the flag will only be set if additionally in case (a), we have that there is a non-trivial attack when testing sID with peer $scc[holder[sID],\pi]$ and in case (b) there is a non-trivial attack with peer $prd[holder[sID],\pi]$. All these conditions map to a specific non-trivial attack that transfers to an underlying attack on UAE_{PFS} : if there is a non-trivial attack on $GAKE_{PFS}$ under these conditions we have that any of the lines (0), (1), (2), or (3) in Table 1 are fulfilled. We show this by analyzing what – under the conditions i), ii), iii), iv) respectively – a non-trivial attack on the $GAKE$ protocol means. Importantly, we show that under these conditions, attacks on the $GAKE$ protocol will *always* result in an attack on the UAE protocol.

Let us begin by considering the reduction for which we give the complete pseudocode in the full version. We observe that the output distribution of the queries $INITIALIZE$, $KEYGENERATION$, $SESSION_B$, $SESSION_R$, $SESSION_F$, $CORRUPT$, DER , $TEST$ and all outputs of the queries to the random oracle are distributed like in Game 2, except for H_{prkey} . Thus, we need to formally show that $FINALIZE$, H_{prkey} , and $REV-STATE$ are distributed like in Game 2 unless the $GAKE$ attacker breaks the underlying UAE security game.

Let us begin with $REV-STATE$. The introduced changes will complete the state of the $GAKE$ with the state of the underlying UAE if needed. By the modifications made in the last games, we have ensured that in no other place the state of the underlying UAE is required. Also, observe that the format of the full state of the $GAKE$ protocol means that only the last part of it refers to the underlying UAE. So by appropriate projections, the state $state'[sID]$ can give information on both the underlying UAE state and the remaining state of the $GAKE$.

Now let us have a look at the remaining queries `FINALIZE` and H_{prkey} . The first one makes the underlying UAKE decide whether there has actually been a non-trivial attack on session sID^* . To this end, it calls the `CHECK` query of the UAKE. We will in the following detail which subset of the attacks on the GAKE will correspond to an attack on the UAKE by comparing the attack tables that both schemes use. Next, `FINALIZE` simply calls the `FINALIZE` query of the UAKE to essentially relay the `attFound` value of the underlying UAKE to the `attFound` of the GAKE. Similarly, H_{prkey} also uses the `CHECK` query of the UAKE to identify if the input session key is consistent with some session sID . Let us begin our analysis of the conditions in the attack tables: first, we consider all GAKE attacks that would imply an attack on the UAKE according to line (0) in Table 1. Such an attack happens if, in any UAKE run, there is a receiver session such that there are multiple initiator sessions that share the first message with the receiver (partially matching sessions). However, since we have already excluded collisions in the first message due to its high entropy this is impossible.

Let us now consider Table 3. Consider lines (0) to (8). The lines all imply that there are two neighbors that have matching sessions since a partially matching session in GAKE implies a fully matching session in the underlying UAKE. Since the underlying UAKE has high entropy, we have that the GAKE attack of line (0) can be excluded with overwhelming probability. We remark that, for the GAKE to have partially matching sessions, two sessions must agree on the first *two* messages. However, each communication partner participates in generating one of the messages via a contribution of high entropy messages. But, due to the previous modifications of the security game where we excluded collisions, we will not see the same message twice. So we can consider the lines (1) to (8) in the GAKE table that will transfer to sessions of the UAKE that have a single matching session. By definition of non-trivial GAKE attacks, we have that for both, the tested session and the partnered sessions, it must hold that state and long-term key will never be revealed at the same time. Both scenarios can be simulated efficiently without immediately presenting a trivial attack according to the UAKE definition. The PFS notion guarantees that we cover corruptions of the long-term key. (For this, WFS would actually be sufficient at this point in the proof.) Importantly now, albeit for non-tested sessions revealing the state of the GAKE can be transferred to revealing states in the UAKE, now the state of the UAKE will never be revealed. The underlying argument is that by revealing the state of the GAKE, the attacker only obtains the encrypted UAKE state $w = H_{\text{st}}(IV, k) + \text{st}$ (or $\hat{w} = H_{\hat{\text{st}}}(I\hat{V}, k) + \hat{\text{st}}$ or $\bar{w} = H_{\bar{\text{st}}}(I\bar{V}, k) + \bar{\text{st}}$). Moreover, the adversary can only obtain (with overwhelming probability) the current underlying UAKE state (st , $\hat{\text{st}}$, or $\bar{\text{st}}$) at all, by obtaining not only the long-term key k via a `CORRUPT` query but also the initialization vector (IV , $I\hat{V}$, or $I\bar{V}$) via a `REV-STATE` query. We emphasize that by the changes introduced in the previous games, there is indeed no other way for the attacker to obtain the UAKE state: since it is modeled as a random oracle, the output of H_{st^*} does not reveal anything about its inputs. It is drawn as a uniformly random output and independent of any other values. Additionally, k will never be used for any

other purpose than computing encrypted states. As it is part of the long-term key it will never be accessible via state revealing. The initialization vectors will likewise never be used for any other purpose besides state encryption. Moreover, they will only be stored in the state of sessions and are not part of the long-term key.

Now let us consider the remaining lines in the GAKE table, (9) and (10). These lines indicate no partially matching session exists for the tested session. However, since the underlying UAKE protocol has only two moves, this can either transfer to the existence of a partially matching session in the UAKE table—line (2)—or the lack of it—line (3). We note that in both cases we can exploit the PFS property of the UAKE in case the attacker corrupts the holder of the test session.

Now consider lines (9) and (10) more closely and let us restrict our attention to the sub-case where in the resulting UAKE table we have partially matching conversations, i.e. the first message has not been modified on transit. With the same arguments as before on state encryption, the conditions in the GAKE table for both lines guarantee that the attacker will never see the secret state of any of the sessions. This in particular holds for the computations of each GAKE session that correspond to the computations of the responder in the underlying UAKE protocol. So this particular sub-case of attacks on the GAKE protocol will always transfer to an attack according to line (2).

Finally, consider the remaining sub-class of attacks that are characterised by the lack of sessions that do not have partially matching sessions with the tested session in the underlying UAKE, i.e. they do not share the first message. The last line of the UAKE table essentially says that in case no oracle shares the first message with the tested oracle, then all other oracles might as well be fully under the control of the attacker. We do not require the peer to be corrupted and have no other requirements on the remaining oracles (since they are not fully nor partially matching the tested oracle). Also, the two lines in the GAKE table make sure that there is no corruption of the holder of the tested session at all. So in particular, there is no “early” corruption where $\text{peersCorr}[\text{sID}^*]$ is true. Due to the state encryption this immediately guarantees that the UAKE state of the tested session is never revealed. Thus, this sub-case transfers to the sub-case (3) of the UAKE table.

To win, the adversary must successfully guess a crucial input to the computation of $K_{\text{UAKE},i}$ or $K'_{\text{UAKE},i}$ and thus would be able to compute these keys directly. In particular, the attacker has generated an input to the random oracle H_{prkey} that constitutes a valid attack. This now immediately reduces to the OW-UAKE-G security game.

$$|\Pr[G_{3,b}^A \Rightarrow 1] - \Pr[G_{2,b}^A \Rightarrow 1]| \leq \Pr[\text{BAD}] \leq \text{Adv}_{\text{UAKE}_{\text{PFS}}}^{\text{PFS}}(\mathcal{B}).$$

GAME $G_{4,b}$. We raise flag BAD_U and abort if there exists a session sID such that a (prime) predecessor UAKE key $K'_{\text{UAKE},\text{prd}[\text{holder}[\text{sID}],\pi]}$ or a (prime) successor UAKE key $K'_{\text{UAKE},\text{holder}[\text{sID}]}$ have been queried (in a valid attack) with the

predecessor or, respectively, the successor as peer. The probability that BAD_U is raised for a specific derived key is at most $q_{H_{\text{ukey}}}/2^\tau$, where $q_{H_{\text{ukey}}}$ are queries issued to random oracle H_{ukey} and $q_{H_{\text{ukey}}} \leq q$. An union bound, over the number of sessions, gives us

$$|\Pr[G_{4,b}^A \Rightarrow 1] - \Pr[G_{3,b}^A \Rightarrow 1]| \leq \Pr[\text{BAD}_U] \leq \frac{Sq_{H_{\text{ukey}}}}{2^\tau}.$$

GAME $G_{5,b}$. We raise flag BAD_T and abort if there is a session sID such that a derived predecessor key $K_{\text{prd}[\text{holder}[\text{sID}],\pi],\text{holder}[\text{sID}]}$ is issued in a valid attack. The probability that BAD_T is raised for a specific $K_{\text{prd}[\text{holder}[\text{sID}],\pi],\text{holder}[\text{sID}]}$ is at most $q_{H_{\text{tag}}}/2^\tau$, where $q_{H_{\text{tag}}}$ are queries issued to random oracle H_{tag} and $q_{H_{\text{tag}}} \leq q$. Again, an union bound gives us

$$|\Pr[G_{5,b}^A \Rightarrow 1] - \Pr[G_{4,b}^A \Rightarrow 1]| \leq \Pr[\text{BAD}_T] \leq \frac{Sq_{H_{\text{tag}}}}{2^\tau}.$$

GAME $G_{6,b}$. We raise flag BAD_G and abort if there exists a session sID such that a sorted key \bar{K} is issued in a valid attack for all the peers of the session. The probability that BAD_G is raised for a specific \bar{K} is at most $q_{H_{\text{gkey}}}/|\mathcal{K}_{\text{GAKE}_{\text{PFS}}}|$, where $q_{H_{\text{gkey}}}$ are queries issued to random oracle H_{gkey} and $q_{H_{\text{gkey}}} \leq q$. An union bound gives us

$$|\Pr[G_{6,b}^A \Rightarrow 1] - \Pr[G_{5,b}^A \Rightarrow 1]| \leq \Pr[\text{BAD}_G] \leq \frac{Sq_{H_{\text{gkey}}}}{|\mathcal{K}_{\text{GAKE}_{\text{PFS}}}|}.$$

Now, observe that the attacker does not have any advantage in distinguishing the session key from a random key since for all non-trivial attacks the session key is a random key by the modifications that we made in the sequence of games.

Combining all probabilities, we obtain the bound stated in Theorem 2. \square

6 Final GAKE Protocol

A pseudocode description of our final GAKE protocol based on KEMs is given in the full version. The idea is the same showed previously in Fig. 1. It is a multi-party three round protocol.

6.1 UAKE from KEMs

We introduce the syntax for key encapsulation mechanisms and provide the definitions of correctness and min-entropy. The latter is extremely useful for the initial step of the security proof of the UAKE based on KEMs.

SYNTAX. A key encapsulation mechanism $\text{KEM} := (\text{Setup}, \text{KeyGen}, \text{Encaps}, \text{Decaps})$ consist of four polynomial-time algorithms:

- $\text{par} \leftarrow \text{Setup}(1^\kappa)$: The probabilistic setup algorithm Setup takes as input the security parameter κ in unary and returns global system parameters par that implicitly define ciphertext space \mathcal{C} , the public key space \mathcal{PK} , the secret key space \mathcal{SK} and the key space \mathcal{K} .
- $(\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{par})$: The probabilistic key generation algorithm KeyGen takes as input the parameters par and returns a public key $\text{pk} \in \mathcal{PK}$ and a secret key $\text{sk} \in \mathcal{SK}$.
- $(\text{ct}, K) \leftarrow \text{Encaps}(\text{pk})$: The probabilistic encapsulation algorithm Encaps takes as input a public key pk and returns a ciphertext $\text{ct} \in \mathcal{C}$ and a key $K \in \mathcal{K}$.
- $K := \text{Decaps}(\text{sk}, \text{ct})$: The deterministic decapsulation algorithm Decaps takes as input a secret key sk and a ciphertext ct and returns a key $K \in \mathcal{K}$.

Definition 10 (Correctness KEM). *We say that KEM is ρ -correct, if for any $\text{par} \leftarrow \text{Setup}(1^\kappa)$ we have:*

$$\Pr[K = K' \mid (\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{par}), (\text{ct}, K) \leftarrow \text{Encaps}(\text{pk}), K' = \text{Decaps}(\text{sk}, \text{ct})] \geq \rho.$$

Definition 11 (* KEM). *We say that KEM has min-entropy μ if:*

- *It has key min-entropy $\mu' \geq \mu$: for all $\text{pk}' \in \mathcal{PK}$ we have $\Pr[\text{pk} = \text{pk}' : (\text{pk}, \text{sk}) \leftarrow \text{KeyGen}(\text{par})] \leq 2^{-\mu'}$ for some par .*
- *It has ciphertext min-entropy $\mu'' \geq \mu$: for all $\text{ct}' \in \mathcal{C}$ we have $\Pr[\text{ct} = \text{ct}' : (\text{ct}, K) \leftarrow \text{Encaps}(\text{pk})] \leq 2^{-\mu''}$ for some $\text{pk} \in \mathcal{PK}$.*

SECURITY NOTION FOR KEM. We recall the security notion recently used to analyze two-party key exchange with key confirmation from [26], which is a multi-user version of one-way security with corruptions under plaintext checking and ciphertext validity attacks.

Definition 12. *The game OW-PCVA-CR is defined as in Fig. 11. The advantage of an adversary \mathcal{A} against KEM in this game is defined as*

$$\text{Adv}_{\text{KEM}}^{\text{OW-PCVA-CR}}(\mathcal{A}) := \Pr[\text{OW-PCVA-CR}^{\mathcal{A}} \Rightarrow 1].$$

UAKE CONSTRUCTION. We construct a UAKE protocol from two KEMs as shown in Fig. 12. Let us first prove correctness.

Lemma 3. *Consider the construction in Fig. 10. If both KEM^e and KEM^L have overwhelming correctness of at least $\rho = 1 - 1/2^v$ for some $v \in \Omega(\kappa)$ and the attacker makes q queries overall, then UAKE_{WFS} has overwhelming correctness at least $1 - q/2^{v-2}$.*

Proof. We give a crude bound. The proof closely follows the proof of the main construction. Assume each of the two KEM schemes has at least overwhelming correctness $\rho = 1 - 1/2^v$ for some $v \in \Omega(\kappa)$. Observe that by setup the KEMs are the only source of incorrectness in the entire protocol construction. Next, observe that if we condition the KEMs to have no correctness error at all, then we will not have a correctness error in UAKE_{WFS} as well. So we only have to analyse the

GAME OW-PCVA-CR	$\text{REV}(i, c)$
<u>INITIALIZE</u>	12 if $\exists K$ s. t. $(i, c, K) \in \mathcal{L}_{\text{ENC}}$:
00 $(N, \mathcal{L}_{\text{ENC}}, \mathcal{L}_{\text{CORR}}, \mathcal{L}_{\text{REV}}) := (0, \emptyset, \emptyset, \emptyset)$	13 $\mathcal{L}_{\text{REV}} := \mathcal{L}_{\text{REV}} \cup \{(i, c)\}$
01 par \leftarrow Setup	14 return K
02 return par	15 return \perp
<u>KEYGENERATION(par)</u>	$\text{ENC}(i)$
03 $N ++$	16 $(c, K) \leftarrow \text{Encaps}(\text{pk}_i)$
04 $(\text{pk}_N, \text{sk}_N) \leftarrow \text{KeyGen}(\text{par})$	17 $\mathcal{L}_{\text{ENC}} := \mathcal{L}_{\text{ENC}} \cup \{(i, c, K)\}$
05 return pk_N	18 return c
<u>FINALIZE(i^*, c^*, K^*)</u>	$\text{CVO}(i, c')$
06 if $(i^*, c^*, K^*) \notin \mathcal{L}_{\text{ENC}}$: return 0	19 if $\exists K'$ s. t. $(i, c', K') \in \mathcal{L}_{\text{ENC}}$: return \perp
07 if $i^* \in \mathcal{L}_{\text{CORR}}$: return 0	20 $K' := \text{Decaps}(\text{sk}_i, c')$
08 if $(i^*, c^*) \in \mathcal{L}_{\text{REV}}$: return 0	21 return $\llbracket K' \in \mathcal{K} \rrbracket$
09 return 1	
<u>CORR(i)</u>	$\text{CHECK}(i, c, K)$
10 $\mathcal{L}_{\text{CORR}} := \mathcal{L}_{\text{CORR}} \cup \{i\}$	22 return $\llbracket \text{Decaps}(\text{sk}_i, c) = K \rrbracket$
11 return sk_i	

Fig. 11. Game OW-PCVA-CR for KEM. \mathcal{A} has access to oracles $\mathcal{O} := \{\text{ENC}, \text{CVO}, \text{REV}, \text{CHECK}, \text{CORR}\}$.

<u>Setup(1^κ)</u>	<u>Der_R(sk, M_1)</u>
00 par ^e \leftarrow Setup ^e (1^κ)	10 parse $(\text{pk}^e, c^L) := M_1$
01 par ^L \leftarrow Setup ^L (1^κ)	11 $(c^e, K^e) \leftarrow \text{Encaps}^e(\text{pk}^e)$
02 return par := (par ^e , par ^L)	12 $K^L := \text{Decaps}^L(\text{sk}^L, c^L)$
<u>KeyGen(par)</u>	13 $K := (K^e, K^L)$
03 $(\text{pk}^L, \text{sk}^L) \leftarrow \text{KeyGen}^L(\text{par}^L)$	14 $M_2 := c^e$
04 return (pk, sk) := (pk ^L , sk ^L)	15 return (M_2, K)
<u>Beg(pk)</u>	<u>Der_B(pk, M_2, st)</u>
05 $(\text{pk}^e, \text{sk}^e) \leftarrow \text{KeyGen}^e(\text{par}^e)$	16 parse $(\text{sk}^e, K^L) := \text{st}$
06 $(c^L, K^L) \leftarrow \text{Encaps}^L(\text{pk}^L)$	17 $K^e := \text{Decaps}^e(\text{sk}^e, M_2)$
07 $M_1 := (\text{pk}^e, c^L)$	18 $K := (K^e, K^L)$
08 st := (sk ^e , K ^L)	19 return K
09 return (M_1 , st)	

Fig. 12. Generic construction of UAKE_{WFS} from KEM^e and KEM^L .

influence of the KEMs on the overall correctness. Now, since a single application of a single KEM has overwhelmingly high correctness ρ the probability for both KEMs to simultaneously have no correction error is lower bounded by $\rho' = \rho^2 \geq 1 - 1/2^{v-2}$. A q -time call of the KEMs will thus result in a correctness of $(1 - 1/2^{v-2})^q$. This can be lower bounded via $(1 - 1/2^{v-2})^q \geq (1 - q/2^{v-2})$ for some arbitrary polynomial $q = q(\kappa)$ which shows that the resulting correctness is still overwhelming. \square

Theorem 3 (KEM to UAKE_{WFS}). *For any adversary \mathcal{A} against OW-UAKE- G_{WFS} with protocol UAKE_{WFS} with N parties that establish at most S sessions and issues at*

most q queries to the oracles, there exist adversaries \mathcal{B} and \mathcal{C} against OW-PCVA-CR security of KEM^e and KEM^L with respectively min-entropy μ^e and μ^L and correctness $(1 - 1/2^v)$ for both, such that

$$\begin{aligned} \text{Adv}_{\text{UAKE}_{\text{WFS}}}^{\text{WFS}}(\mathcal{A}) \leq & \text{Adv}^{\text{OW-PCVA-CR}}_{\text{KEM}^e}(\mathcal{B}) + \text{Adv}^{\text{OW-PCVA-CR}}_{\text{KEM}^L}(\mathcal{C}) \\ & + \frac{2S^2}{2^{\mu^e}} + \frac{N^2 + S^2}{2^{\mu^L}} + \frac{q}{2^{v-2}}, \end{aligned}$$

where the running time of \mathcal{B} and \mathcal{C} is about that of \mathcal{A} .

We provide the proof in the full version.

6.2 Putting Things Together

We collect the bounds from Theorems 1 to 3 in the following corollary.

Corollary 1 (KEMs to GAKE). *For any adversary \mathcal{A} against $\text{GAKE-G}_{\text{PFS}}$ with protocol GAKE_{PFS} with N parties that establish at most S sessions and issues at most q queries to the oracles, there exist adversaries \mathcal{B} and \mathcal{C} against OW-PCVA-CR security of KEM^e and KEM^L with respectively min-entropy μ^e and μ^L and correctness $(1 - 1/2^v)$ for both, such that*

$$\begin{aligned} \text{Adv}_{\text{GAKE}_{\text{PFS}}}^{\text{PFS}}(\mathcal{A}) \leq & \text{Adv}^{\text{OW-PCVA-CR}}_{\text{KEM}^e}(\mathcal{B}) + \text{Adv}^{\text{OW-PCVA-CR}}_{\text{KEM}^L}(\mathcal{C}) \\ & + \frac{Sq_{RS} + Nq_C + N^2 + 3S^2}{2^\kappa} + \frac{3Sq_{RO} + 2q_{RO}^2 + S + q_{Ch}}{2^\zeta} \\ & + \frac{3(N^2 + S^2)}{2^{\mu^L}} + \frac{2S^2 + Sq_{RO}}{2^{\mu^e}} + \frac{5q + 2S}{2^v}, \end{aligned}$$

where ζ is the lower bound for all dimensions of the random oracle outputs and $q_{RO}, q_{Ch}, q_{RS}, q_C$ are the number of random oracle, check, reveal state and corrupt queries that \mathcal{A} makes. Further, the running time of \mathcal{B} and \mathcal{C} are about that of \mathcal{A} .

INSTANTIATIONS FROM DDH AND LWE. Let us now show how we can implement the underlying KEM with existing constructions. The first implementation is the DDH-based KEM introduced in [19]. As shown in [26, 28] this scheme achieves OW-PCVA-CR security. Ciphertexts consist of two group elements, whereas public keys consist of a single group element. Overall we thus need to transfer two ciphertext and one ephemeral public key in the UAKE protocol with WFS. The UAKE protocol with PFS adds to this a MAC which accounts for a bitstring of size 256 bits. Finally, in the second phase, we exchange symmetric ciphertexts of size 256 bits. In total, this accounts for 5 group elements and 2 strings of size 256 bits.

In comparison, the tightly secure protocol by [25] requires to send 2 group elements for the underlying BD protocol. They use signature schemes over each BD message for authentication. Unfortunately, at the time their paper was published

no efficient signature scheme was known that fulfilled the security notion that they require in a tightly secure way. This is why they used Schnorr signatures while providing a proof of tight security *in the generic group model*. However, at the same time, the work in [13] shows that Schnorr signatures cannot provide tight security under any non-interactive security assumption. This indicates that the results of [25] will lose its tight security guarantees when leaving the GGM model and reducing to non-interactive security assumptions. Based on this instantiation, their protocol accounts for overall 2 group elements and 4 exponents in \mathbb{Z}_p where p is the group order. However, the recent signature scheme in [10] can now be used as a drop-in for their protocol to obtain a proof under the DDH assumption that does not rely on the generic group model. This signature scheme has signatures that consist of 3 elements in \mathbb{Z}_p . When using it in [25] this accounts for 6 elements in \mathbb{Z}_p and 2 group elements. We can now instantiate all schemes in elliptic curve groups with group element representation of around 256 bits.

As a result, we can see that our protocol is more efficient when reducing to non-interactive security assumptions and not relying on GGM proofs. At the same time, we stress that we prove security in a much stronger model that allows the attacker to reveal secret states.

Our second implementation uses the recent scheme by [28] that is based on the LWE assumption. It is thus secure in the PQ-setting. The construction relies on a double encryption approach and is given the full version along with an analysis that shows that its correctness bounds are suitable for our transformation.

Acknowledgements. Emanuele Di Giandomenico and Sven Schäge have been supported by the CONFIDENTIAL6G project that is co-funded by the European Union (grant agreement ID: 101096435). Work done while Doreen Riepel was at UC San Diego, supported in part by Mihir Bellare’s KACST grant.

References

1. Abdalla, M., Bohli, J.M., González Vasco, M.I., Steinwandt, R.: (Password) authenticated key establishment: From 2-party to group. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 499–514. Springer, Berlin, Heidelberg (Feb 2007). https://doi.org/10.1007/978-3-540-70936-7_27
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, Cham (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, Cham (Nov 2020). https://doi.org/10.1007/978-3-030-64378-2_10
4. Apon, D., Dachman-Soled, D., Gong, H., Katz, J.: Constant-round group key exchange from the ring-LWE assumption. In: Ding, J., Steinwandt, R. (eds.) Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019. pp. 189–205. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25510-7_11




5. 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, Berlin, Heidelberg (May 2016). https://doi.org/10.1007/978-3-662-49896-5_10
6. Bienstock, A., Dodis, Y., Garg, S., Grogan, G., Hajiabadi, M., Rösler, P.: On the worst-case inefficiency of CGKA. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part II. LNCS, vol. 13748, pp. 213–243. Springer, Cham (Nov 2022). https://doi.org/10.1007/978-3-031-22365-5_8
7. Burmester, M., Desmedt, Y.: A secure and efficient conference key distribution system (extended abstract). In: Santis, A.D. (ed.) EUROCRYPT’94. LNCS, vol. 950, pp. 275–286. Springer, Berlin, Heidelberg (May 1995). <https://doi.org/10.1007/BFb0053443>
8. Cohn-Gordon, K., Cremers, C.: Mind the gap: Where provable security and real-world messaging don’t quite meet. Cryptology ePrint Archive, Report 2017/982 (2017), <https://eprint.iacr.org/2017/982>
9. Cohn-Gordon, K., Cremers, C., Garratt, L., Millican, J., Milner, K.: On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 1802–1819. ACM Press (Oct 2018). <https://doi.org/10.1145/3243734.3243747>
10. 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, Cham (May 2021). https://doi.org/10.1007/978-3-030-75248-4_1
11. Dodis, Y., Fiore, D.: Unilaterally-authenticated key exchange. In: Kiayias, A. (ed.) FC 2017. LNCS, vol. 10322, pp. 542–560. Springer, Cham (Apr 2017). https://doi.org/10.1007/978-3-319-70972-7_31
12. Dutta, R., Barua, R.: Constant round dynamic group key agreement. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 74–88. Springer, Berlin, Heidelberg (Sep 2005). https://doi.org/10.1007/11556992_6
13. Fleischhacker, N., Jager, T., Schröder, D.: On tight security proofs for Schnorr signatures. In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014, Part I. LNCS, vol. 8873, pp. 512–531. Springer, Berlin, Heidelberg (Dec 2014). https://doi.org/10.1007/978-3-662-45611-8_27
14. Gorantla, M.C., Boyd, C., González Nieto, J.M.: Modeling key compromise impersonation attacks on group key exchange protocols. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 105–123. Springer, Berlin, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00468-1_7
15. 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, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84259-8_23
16. 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, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_22
17. Hövelmanns, K., Kiltz, E., Schäge, S., Unruh, D.: Generic authenticated key exchange in the quantum random oracle model. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part II. LNCS, vol. 12111, pp. 389–422. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45388-6_14

18. Ishibashi, R., Yoneyama, K.: Post-quantum anonymous one-sided authenticated key exchange without random oracles. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022, Part II. LNCS, vol. 13178, pp. 35–65. Springer, Cham (Mar 2022). https://doi.org/10.1007/978-3-030-97131-1_2
19. 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, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_5
20. Klein, K., Pascual-Perez, G., Walter, M., Kamath, C., Capretto, M., Cueto, M., Markov, I., Yeo, M., Alwen, J., Pietrzak, K.: Keep the dirt: Tainted TreeKEM, adaptively and actively secure continuous group key agreement. In: 2021 IEEE Symposium on Security and Privacy. pp. 268–284. IEEE Computer Society Press (May 2021). <https://doi.org/10.1109/SP40001.2021.00035>
21. Krawczyk, H.: HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 546–566. Springer, Berlin, Heidelberg (Aug 2005). https://doi.org/10.1007/11535218_33
22. LaMacchia, B.A., Lauter, K., Mityagin, A.: Stronger security of authenticated key exchange. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) ProvSec 2007. LNCS, vol. 4784, pp. 1–16. Springer, Berlin, Heidelberg (Nov 2007). https://doi.org/10.1007/978-3-540-75670-5_1
23. Maurer, U., Tackmann, B., Coretti, S.: Key exchange with unilateral authentication: Composable security definition and modular protocol design. Cryptology ePrint Archive, Report 2013/555 (2013), <https://eprint.iacr.org/2013/555>
24. Mayer, A.J., Yung, M.: Secure protocol transformation via “expansion”: From two-party to groups. In: Motiwalla, J., Tsudik, G. (eds.) ACM CCS 99. pp. 83–92. ACM Press (Nov 1999). <https://doi.org/10.1145/319709.319721>
25. Pan, J., Qian, C., Ringerud, M.: Signed (group) Diffie-Hellman key exchange with tight security. *Journal of Cryptology* **35**(4), 26 (Oct 2022). <https://doi.org/10.1007/s00145-022-09438-y>
26. Pan, J., Riepel, D., Zeng, R.: Key exchange with tight (full) forward secrecy via key confirmation. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VII. LNCS, vol. 14657, pp. 59–89. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58754-2_3
27. Pan, J., Wagner, B.: Lattice-based signatures with tight adaptive corruptions and more. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022, Part II. LNCS, vol. 13178, pp. 347–378. Springer, Cham (Mar 2022). https://doi.org/10.1007/978-3-030-97131-1_12
28. 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, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_20
29. Pan, J., Wagner, B., Zeng, R.: Tighter security for generic authenticated key exchange in the QROM. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part IV. LNCS, vol. 14441, pp. 401–433. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8730-6_13
30. Poettering, B., Rösler, P., Schwenk, J., Stebila, D.: SoK: Game-based security models for group key exchange. In: Paterson, K.G. (ed.) CT-RSA 2021. LNCS, vol. 12704, pp. 148–176. Springer, Cham (May 2021). https://doi.org/10.1007/978-3-030-75539-3_7

31. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th FOCS. pp. 124–134. IEEE Computer Society Press (Nov 1994). <https://doi.org/10.1109/SFCS.1994.365700>
32. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004), <https://eprint.iacr.org/2004/332>



Anamorphic Authenticated Key Exchange: Double Key Distribution Under Surveillance

Weihao Wang^{1,2} , Shuai Han^{1,2} , and Shengli Liu^{2,3} 

¹ School of Cyber Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

{dykler123,dalen17}@sjtu.edu.cn

² State Key Laboratory of Cryptology, P.O. Box 5159, Beijing 100878, China

³ Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China

s1liu@sjtu.edu.cn

Abstract. Anamorphic encryptions and anamorphic signatures assume a double key pre-shared between two parties so as to enable the transmission of covert messages. How to securely and efficiently distribute a double key under the dictator's surveillance is a central problem for anamorphic cryptography, especially when the users are forced to surrender their long-term secret keys or even the randomness used in the algorithms to the dictator.

In this paper, we propose Anamorphic Authentication Key Exchange (AM-AKE) to solve the problem. Similar to anamorphic encryption, AM-AKE contains a set of anamorphic algorithms besides the normal algorithms. With the help of the anamorphic algorithms in AM-AKE, the initiator and the responder are able to exchange not only a session key but also a double key. We define robustness and security notions for AM-AKE, and also prove some impossibility results on plain AM-AKE whose anamorphic key generation algorithm only outputs a key-pair. To bypass the impossibility results, we work on two sides.

- On the one side, for plain AM-AKE, the securities have to be relaxed to resist only passive attacks from the dictator. Under this setting, we propose a generic construction of two-pass plain AM-AKE from a two-pass AKE with partially randomness-recoverable algorithms.
- On the other side, we consider (non-plain) AM-AKE whose key generation algorithm also outputs an auxiliary trapdoor besides the key-pairs. We ask new properties from AKE: its key generation algorithm has secret extractability and other algorithms have separability. Based on such a two-pass AKE, we propose a generic construction of two-pass (non-plain) AM-AKE. The resulting AM-AKE enjoys not only robustness but also the strong security against any dictator knowing both users' secret keys and even the internal randomness of the AKE algorithms and implementing active attacks.

Finally, we present concrete AM-AKE schemes from the popular SIG+KEM paradigm and three-KEM paradigm for constructing AKE.

1 Introduction

Cryptography provides fundamental technical tools for achieving authenticity and confidentiality in our daily electronic data communications. For a cryptographic algorithm to work, it is critical that the underlying secret key is not compromised by the adversary. However, an authority dictator may force citizens to surrender their secret keys, and as a result, cryptographic algorithms may completely lose their functionalities of authenticity and confidentiality.

To save cryptographic functionalities in face of dictator, the so-called anamorphic algorithms were introduced [1, 5, 13, 20, 25].

Anamorphic Algorithms Supported by Double Key. In [20], Persiano, Phan, and Yung proposed the concept of anamorphic encryption (AME), which is partitioned into receiver-AME and sender-AME depending on whether the receivers are forced to surrender their secret keys or the senders are forced to send designated messages. As for receiver-AME, it is a public-key encryption (PKE) deployed either in normal model with (Gen, Enc, Dec) or in the anamorphic mode with (aGen, aEnc, aDec). In the anamorphic mode, the receiver initially generates an anamorphic key-pair (ask, apk) and a *double key* dk via aGen. Any sender who shares dk with the receiver is able to use apk and dk to encrypt not only a normal plaintext m but also a covert plaintext \hat{m} . The anamorphic public key apk and the resulting anamorphic ciphertext \hat{c} should be indistinguishable from the normal public key pk and normal ciphertext c to the dictator who also obtains the corresponding secret key. With the knowledge of secret key, the dictator can always decrypt the ciphertext to learn m , but the covert message \hat{m} remains hidden owing to the secrecy of the double key dk .

Later, Kutyłowski et al. extended anamorphic encryption to anamorphic signature [13], where a signing party can use the anamorphic signing key ask and the double key dk to send undetectable secure messages using signature tags which are indistinguishable from regular tags for the dictator who sees the signing key ask but not the double key dk .

The original anamorphic schemes bind dk to the anamorphic key pair (apk, ask). Once the public key is deployed, it is not possible to associate the public key with a double key. This limitation is lifted in [1] by allowing double keys to be created independently of key-pairs, which makes it possible to create double keys at anytime even after the public key is deployed.

Double Key Distribution. The double key dk is essential to anamorphic encryption and anamorphic signature, whose security relies on the secrecy of dk (to the dictator). Now the crucial problem is how to secretly distribute the double key dk between sender and receiver in face of the dictator who may obtain the secret key of all users. The offline physical delivery of dk is expensive and even infeasible in the Internet era. In [20], a two-step bootstrap method in [11] was suggested for distributing dk secretly: Superficially, two parties send to each other abundant ciphertexts generated by a PKE scheme. Covertly, they implement a key-exchange (KE) protocol. Each ciphertext from PKE embeds a tiny piece of the pseudo-random transcript of KE. If they can collect the complete

transcript of KE, they can compute a common dk . This method is very inefficient, since its embedding rate is very low. Even worse, this method is too fragile to be practical, since the parties have to collect all these ciphertexts (e.g., hundreds or even thousands of ciphertexts) to recover the KE transcripts, and any active attack or transmission disordering will ruin the distribution of dk . Another possible way for double key distribution might be via sender-AME [25]. However, sender-AME does not allow the dictator to obtain the users' secret keys, which is not compatible to the security settings considered by other anamorphic schemes like receiver-AME or anamorphic signature, and thus this method seems hardly useful for distributing double keys for those anamorphic schemes. Therefore, a natural and important question is:

How to distribute double keys dk in a secure and efficient way under the surveillance of the dictator?

Our answer to the question is *Anamorphic Authenticated Key Exchange*.

1.1 Our Contributions

In this paper, we initiate the study of Anamorphic Authenticated Key Exchange (AM-AKE) and formalize security requirements for it, including robustness, indistinguishability of working modes (IND-WM) and pseudo-randomness of double keys (PR-DK). Then we provide impossibility results and possibility results on achieving secure AM-AKE. In particular, we show that two popular paradigms for constructing AKE are good candidates for obtaining AM-AKE: the first one is the signed Diffie-Hellman paradigm [17] which uses a digital signature scheme (SIG) and a key encapsulation mechanism (KEM), referred to as the SIG+KEM paradigm in this paper, and the second one is the three-KEM paradigm [18] which invokes KEM three times. Actually, many existing AKE schemes are designed following these paradigms, such as the IKE protocol [10], the protocol used in TLS 1.3 [21], the 2KEM+Diffie-Hellman protocol [4] and more in [7, 9, 12, 15, 19, 26]. For efficiency consideration, we focus on two-pass AM-AKE.

Syntax, Robustness and Security Notions of AM-AKE. We define two-pass AM-AKE with AKE's normal algorithms and a set of anamorphic algorithms. With the normal algorithms, a session key K is agreed between the initiator and the responder. With the anamorphic ones, both a session key K and a double key dk are agreed between the initiator and the responder.

To make AM-AKE useful, we define initiator-robustness (resp., responder-robustness) as the initiator's (resp., responder's) capability of telling whether its partner is working with the normal or anamorphic algorithms. The robustness helps the party invalidate its double key when its partner works with normal algorithms (i.e., its partner has no intention to share any double key).

As for security, we define indistinguishability between parties' different working modes (IND-WM security) against the dictator who possesses the secret keys of all the parties, the session keys, and even the states of the initiator in any completed AKE sessions, and is permitted to conduct active attacks. This ensures

that the dictator cannot realize that the parties are actually invoking anamorphic algorithms to establish double keys. For such a dictator, we also define pseudo-randomness of the double keys (PR-DK security) to capture that the dictator learns no information about the double keys. This guarantees that the pseudo-random double keys can be later used in anamorphic encryption or signature schemes to transmit covert messages.

We also consider *strong security* notions of IND-WM and PR-DK, denoted by sIND-WM and sPR-DK respectively. sIND-WM and sPR-DK are defined similar to IND-WM and PR-DK, but they deal with a stronger dictator who not only implements passive and active attacks, obtains secret keys of the parties, the session keys and the internal states, but also forces the parties to surrender their internal randomness used in AKE sessions.

Impossibility Results for Plain AM-AKE. For a *plain* AM-AKE where the output of the anamorphic key generation algorithm aGen only contains an anamorphic key-pair $(\mathsf{apk}, \mathsf{ask})$, we prove three impossibility results.

- It’s impossible for a two-pass plain AM-AKE to achieve responder-robustness.
- It’s impossible for a plain AM-AKE to achieve both initiator-robustness and IND-WM security, due to the dictator’s active attacks of impersonating the initiator with its secret key to test the working mode of its partner.
- It’s impossible for a plain AM-AKE to achieve PR-DK security under active attacks, since the dictator can impersonate any party with its secret key to agree on a double key.

Generic Construction of Plain AM-AKE with Relaxed Security. To bypass the impossibility results, we relax the security requirements for plain AM-AKE by restricting the active attacks by adversary. The relaxed IND-WM security excludes the attacks of impersonating the initiator, while the relaxed PR-DK security excludes the attacks of impersonating either the initiator or the responder. Then we propose a generic construction of plain AM-AKE achieving initiator-robustness, the relaxed IND-WM and PR-DK security from any AKE with partially randomness-recoverable algorithms. We prove that those AKE under the SIG+KEM paradigm and those under the three-KEM paradigm are both good candidates, as long as the underlying SIG and/or KEM schemes are randomness-recoverable.

Generic Construction of Robust AM-AKE with Strong Security. Recall that the impossibility results apply to plain AM-AKE, so another possible way of bypassing the impossibility is designing *non-plain* AM-AKE, where the anamorphic key generation algorithm $(\mathsf{apk}, \mathsf{ask}, \mathsf{aux}) \leftarrow \mathsf{aGen}$ outputs a related auxiliary trapdoor aux along with the anamorphic key-pair $(\mathsf{apk}, \mathsf{ask})$. We note that aux is only kept by the party who generates it and does not need to share it with others. In other words, we do not require the parties pre-share any prior information anyway.

To construct such AM-AKE, we require *secret extractability* for aGen which enables the initiator and the responder to agree on a common secret s computed

from the auxiliary trapdoor aux of one party and the public key apk' of the other party. Then we propose a generic construction of AM-AKE achieving the strong IND-WM and strong PR-DK security from any AKE whose algorithm Gen is secret extractable. We prove that those AKE under the SIG+KEM paradigm and those under the three-KEM paradigm are both good candidates, as long as the underlying SIG and/or KEM schemes have secret extractable key generation algorithm.

1.2 Technique Overview

For a two-pass AKE scheme $\text{AKE} = (\text{Gen}, \text{Init}, \text{DerR}, \text{DerI})$, the key generation algorithm Gen returns a key-pair (pk, sk) , the initialization algorithm Init computes the first-pass message msg_i , the derivation algorithm DerR for responder derives the second-pass message msg_r and the session key K_r , and the derivation algorithm DerI for initiator derives the session key K_i . For AM-AKE, it additionally has a set of anamorphic algorithms $(\text{aGen}, \text{aInit}, \text{aDerR}, \text{aDerI})$ for deriving double keys (and session keys as well). Let P_i and P_r denote the initiator and responder respectively.

Impossibility Results for Plain AM-AKE. Roughly speaking, AM-AKE is called a plain one, if the anamorphic key generation algorithm aGen only outputs an anamorphic key-pair (apk, ask) . For a plain AM-AKE, the parties will have no advantage over the dictator who owns their key-pairs as well, leading to the consequence that the dictator can impersonate any party to conduct active attacks. So we have the following impossibility results on plain AM-AKE.

- It's impossible for a two-pass plain AM-AKE to achieve responder-robustness. The responder-robustness means that P_r can decide whether P_i invokes normal algorithms or anamorphic algorithms upon receiving the first-pass message from P_i . Note that the adversary who obtains the secret key of P_r can also make the same judgement, thus distinguishing the working mode of P_i and breaking the security of AM-AKE.
- It's impossible for a plain AM-AKE to achieve both initiator-robustness and IND-WM security. With the secret key of P_i , the adversary can impersonate P_i to generate an anamorphic message amsg_i , send it to P_r , and receive a second-pass message from P_r . Note that the initiator-robustness ensures that the adversary who obtains the secret key of P_i can decide whether P_r invokes normal algorithms or anamorphic algorithms, thus breaking the IND-WM security of AM-AKE.
- It's impossible for a plain AM-AKE to achieve the PR-DK security under active attacks. Similarly, the adversary can impersonate P_i by sending amsg_i to P_r , and compute its double key dk_i upon receiving anamorphic message amsg_r from P_r . Note that the correctness of AM-AKE ensures the consistency of double keys $\text{dk}_i = \text{dk}_r$, and thus the adversary trivially knows P_r 's double key dk_r ($= \text{dk}_i$) and breaks the PR-DK security of AM-AKE.

Generic Construction of Plain AM-AKE with Relaxed Security. Let $\text{AKE} = (\text{Gen}, \text{Init}, \text{DerR}, \text{DerI})$ be a two-pass AKE. Let KE be a two-pass key

exchange scheme, like the Diffie-Hellman protocol [6] with the first message g^a and the second message g^b . In the main body of this paper, this KE is accomplished by a KEM scheme with the pseudo-random KEM public key $\widetilde{\text{pk}}$ as the first message and the pseudo-random ciphertext ψ as the second message.

The anamorphic algorithms (aGen , aInit , aDerR , aDerI) of AM-AKE scheme are almost the same as the normal ones, except that KE's two messages g^a and g^b are used for the (partial) randomnesses to generate AKE's two anamorphic messages $\text{amsg}_i, \text{amsg}_r$:

$$\text{amsg}_i \leftarrow \text{Init}(\underbrace{g^a | \dots}_{\text{randomness}}), \quad (\text{amsg}_r, K_r) \leftarrow \text{DerR}(\text{amsg}_i; \underbrace{g^b | \dots}_{\text{randomness}}),$$

where the public key and secret key are omitted from the input for simplicity.

If Init and DerR are partially randomness-recoverable, which means there are recovering algorithms for P_i and P_r to recover g^a and g^b from amsg_i and amsg_r , respectively, then the double key $\text{dk} := g^{ab}$ is shared between P_i and P_r .

For passive attacks from the dictator, the uniformity of g^a and g^b guarantees the (statistical) indistinguishability between normal algorithm Init and anamorphic algorithm aInit , and the (statistical) indistinguishability between DerR and aDerR . Meanwhile, the DDH assumption guarantees the pseudo-randomness of dk , even if the dictator obtains both P_i and P_r 's secret key and even the underlying randomness ($g^a | \dots$) and ($g^b | \dots$).

The initiator-robustness can be achieved if we replace randomness ($g^b | \dots$) with ($g^b | \sigma := \text{PRF}(g^{ab}, \text{amsg}_i) | \dots$) and set $\text{dk} := \text{PRF}(g^{ab}, \text{amsg}_i | \text{amsg}_r)$ with the help of a PRF. In this case P_i is able to tell the working mode of P_r by testing whether $\sigma = \text{PRF}(g^{ab}, \text{amsg}_i)$.

In fact, lots of AKE constructions support partially randomness-recoverable property. For example, in AKE under the SIG+KEM paradigm [17] and that under the three-KEM paradigm [18], the underlying SIG and KEM have instantiations with randomness-recoverable property [2, 3, 8, 16]. Accordingly, such AKE admits AM-AKE schemes with initiator-robustness and relaxed security.

Generic Construction of Robust AM-AKE with Strong Security. To achieve (strong) IND-WM and PR-DK security and bypass the impossibility results, we allow the anamorphic key generation algorithm $(\text{apk}, \text{ask}, \text{aux}) \leftarrow \text{aGen}$ of AM-AKE outputs a related auxiliary trapdoor aux along with the anamorphic key-pair (apk, ask) . The auxiliary message aux is only kept privately by the party who generates it and does not need to share it with others.

To construct such AM-AKE, we require new properties for the two-pass AKE scheme $\text{AKE} = (\text{Gen}, \text{Init}, \text{DerR}, \text{DerI})$, where Gen has *secret extractability*, and Init and DerR have *separable sub-algorithms*.

Roughly speaking, secret extractability of Gen asks a simulating key generation algorithm SimGen and a secret extracting algorithm Extract satisfying the following properties.

- SimGen outputs not only a key-pair (pk, sk) that is indistinguishable to the output of Gen , but also a master key msk serving as the auxiliary trapdoor.

- $\text{Extract}(\text{msk}_i, \text{pk}_r) = s = \text{Extract}(\text{msk}_r, \text{pk}_i)$ for all $(\text{pk}_i, \text{sk}_i, \text{msk}_i) \leftarrow \text{SimGen}$ and $(\text{pk}_r, \text{sk}_r, \text{msk}_r) \leftarrow \text{SimGen}$. The extracting algorithm can extract a secret s from one party's master key and the other party's public key and make sure that two parties can compute the same secret $s = \text{Extract}(\text{msk}_i, \text{pk}_r) = \text{Extract}(\text{msk}_r, \text{pk}_i)$. The extracted secret s is pseudo-random even in the presence of sk_i and sk_r .

DOUBLE KEY GENERATION. Now let SimGen play the role of aGen to generate the anamorphic key-pair (apk, ask) and the auxiliary trapdoor $\text{aux} := \text{msk}$. Then P_i and P_r use their key-pairs to run the AKE protocol and obtain the two pass messages $(\text{msg}_i, \text{msg}_r)$. At the same time, they can use Extract to compute a common secret $s = \text{Extract}(\text{msk}_i, \text{pk}_r) = \text{Extract}(\text{msk}_r, \text{pk}_i)$, and then use s as the seed of PRF to compute the double key

$$\text{dk}_i = \text{PRF}(s, (\text{amsg}_i, \text{amsg}_r)) = \text{dk}_r.$$

ACHIEVING ROBUSTNESS. To achieve robustness, P_i and P_r need to decide the working mode of each other. Our method is that the party invoking anamorphic algorithms provides a proof and embeds the proof in the message, and the other party extracts the proof from the message and verifies the proof. If the proof is valid, then the other party validates its double key and achieves its robustness.

Let us work on responder-robustness first. We require that the normal algorithm Init can be divided into three sub-algorithms $(f_{1,1}, f_{1,2}, \overline{\text{Init}})$ which computes the three parts of $\text{msg}_i = (m_{i,1}, m_{i,2}, m_{i,3})$ respectively. Here $f_{1,1}, f_{1,2}$ make use of independent randomness $d_{i,1}, d_{i,2}$ to compute $m_{i,1} := f_{1,1}(d_{i,1})$ and $m_{i,2} := f_{1,2}(d_{i,2})$, and $\overline{\text{Init}}$ uses independently chosen randomness $d_{i,3}$ to compute $m_{i,3} := \overline{\text{Init}}(d_{i,1}, d_{i,2}, d_{i,3})$ together with $d_{i,1}, d_{i,2}$. This is captured by the *3-separability of Init*.

If P_i invokes anamorphic algorithm alnit , then P_i can prove it by embedding the PRF value $\text{PRF}(s, m_{i,1})$ in $d_{i,2}$. Then the anamorphic alnit works as follows.

- $\underline{\text{amsg}_i} = (m_{i,1}, m_{i,2}, m_{i,3}) \leftarrow \text{alnit} : m_{i,1} := f_{1,1}(d_{i,1}),$
 $m_{i,2} := f_{1,2}(d_{i,2} = \text{PRF}(s, m_{i,1})), (m_{i,3}, \text{st}) \leftarrow \overline{\text{Init}}(d_{i,1}, d_{i,2}, d_{i,3}).$

Next, upon receiving amsg_i , P_r can check whether $m_{i,2} = f_{1,2}(\text{PRF}(s, m_{i,1}))$. If yes, P_i must have invoked anamorphic algorithm alnit , and P_r will invoke aDerR to output amsg_r and accept its double key $\text{dk}_r = \text{PRF}(s, (\text{amsg}_i, \text{amsg}_r))$, otherwise invalidate it with $\text{dk}_r := \perp$. Note that in the normal mode, a uniform $d_{i,2}$ hardly collides with $\text{PRF}(s, m_{i,1})$. We further require that $f_{1,2}$ returns different outputs on different inputs, which is captured with entropy-preserving property. Then $m_{i,2} := f_{1,2}(d_{i,2})$ with uniform $d_{i,2}$ hardly collides with $f_{1,2}(\text{PRF}(s, m_{i,1}))$. So P_r can always correctly decide whether P_i invokes normal algorithm Init or anamorphic algorithm alnit , and hence achieve responder-robustness.

In the same way, we can achieve initiator-robustness by requiring that the normal algorithm DerR has 3-separability with sub-algorithms $(f_{R,1}, f_{R,2}, \overline{\text{DerR}})$ computing $\text{msg}_r = (m_{r,1}, m_{r,2}, m_{r,3})$ and $f_{R,2}$ has the property of entropy-preserving. More precisely, if P_r invokes anamorphic algorithm aDerR , then P_r

can prove this fact by embedding the PRF value $\text{PRF}(s, (m_{i,1}, m_{r,1}))$ in $d_{r,2}$. Consequently, the anamorphic aDerR works as follows.

- $\text{amsg}_r = (m_{r,1}, m_{r,2}, m_{r,3}) \leftarrow \text{aDerR}(\text{amsg}_i) : m_{r,1} := f_{R,1}(d_{r,1}),$
 $m_{r,2} := f_{R,2}(d_{r,2} = \text{PRF}(s, (m_{i,1}, m_{r,1}))), (m_{r,3}, K_r) \leftarrow \overline{\text{DerR}}(\text{amsg}_i, d_{r,1}, d_{r,2}, d_{r,3}).$

Upon receiving amsg_r , P_i can check whether $m_{r,2} = f_{R,2}(\text{PRF}(s, (m_{i,1}, m_{r,1})))$. If yes, P_r must work in anamorphic mode, and P_i will accept its double key $\text{dk}_i = \text{PRF}(s, (\text{amsg}_i, \text{amsg}_r))$, otherwise invalidate it with $\text{dk}_i := \perp$. Meanwhile, P_i also computes the session key with $K_i \leftarrow \text{Derl}(\text{apk}_r, \text{ask}_i, \text{amsg}_r, \text{st})$. Here the anamorphic aDerl is exactly the normal Derl . With a similar analysis as above, we have initiator-robustness.

ACHIEVING STRONG SECURITY OF IND-WM AND PR-DK. We note that the dictator does not know the auxiliary trapdoors $\text{msk}_i, \text{msk}_r$, and hence the extracted secret s is pseudo-random even if the dictator obtains the key-pairs $(\text{apk}_i, \text{ask}_i)$ and $(\text{apk}_r, \text{ask}_r)$.

Let us first consider strong IND-WM security. The difference between the normal algorithm Init and the anamorphic alnit lies in that a random $d_{i,2} \leftarrow_{\S} \mathcal{D}_{1,2}$ is used in Init while a PRF value $d_{i,2} := \text{PRF}(s, f_{1,1}(d_{i,1}))$ with $d_{i,1} \leftarrow_{\S} \mathcal{D}_{1,1}$ is used in alnit .

Now we require $f_{1,1}$ have the property of entropy-preserving, so different inputs to $f_{1,1}$ will lead to different outputs overwhelmingly. Accordingly, every invocation of alnit will result in fresh $d_{i,1}$ and thus fresh $f_{1,1}(d_{i,1})$. Furthermore, the freshness of $f_{1,1}(d_{i,1})$ makes sure that $d_{i,2} := \text{PRF}(s, f_{1,1}(d_{i,1}))$ is pseudo-random and indistinguishable to $d_{i,2} \leftarrow_{\S} \mathcal{D}_{1,2}$ used in Init . Therefore, P_i 's invoking Init or invoking alnit is indistinguishable to the dictator who knows the secret keys $\text{ask}_i, \text{ask}_r$ and even the randomness $(d_{i,1}, d_{i,2}, d_{i,3})$, and does active attacks with $\text{ask}_i, \text{ask}_r$.

By requiring entropy-preserving property for $f_{R,1}$, we have a similar argument showing that P_r 's invoking DerR or invoking aDerR is indistinguishable to the dictator. We stress that the extracted secret s is pseudo-random to the dictator and the dictator's active attacks with message m to aDerR does not help it to distinguish whether $d_{r,2} = \text{PRF}(s, f_{R,1}(d_{r,1}))$ or $d_{r,2} \leftarrow_{\S} \mathcal{D}_{R,2}$ due to the freshness of $f_{1,1}(d_{r,1})$ and the security of PRF.

Together with the fact that $\text{Derl} = \text{aDerl}$, we know that the AM-AKE has strong indistinguishability of working mode (strong IND-WM) against the dictator. Here "strong" is reflected in that the dictator is able to implement active attacks with secret keys $\text{ask}_i, \text{ask}_r$ and also able to obtain the randomness like $(d_{i,1}, d_{i,2}, d_{i,3})$ and $(d_{r,1}, d_{r,2}, d_{r,3})$.

As for strong PR-DK security, we first consider the dictator's passive attacks, the pseudo-randomness $\text{dk} = \text{PRF}(s, (\text{amsg}_i, \text{amsg}_r))$ is indistinguishable to a random key $\text{dk} \leftarrow_{\S} \mathcal{DK}$, thanks to the freshness of $(\text{amsg}_i, \text{amsg}_r)$ from the entropy-preserving property of $f_{1,1}, f_{1,2}, f_{R,1}, f_{R,2}$. Next we consider the dictator's active attacks with message m . There are two cases.

- (1) This m leads to an invalid double key $\text{dk} = \perp$ (but without s , the dictator does not realize $\text{dk} = \perp$) due to $d_{i,2} \neq \text{PRF}(s, m_{i,1})$ or $d_{r,2} \neq \text{PRF}(s, (m_{i,1}, m_{r,1}))$.
- (2) If $d_{i,2} = \text{PRF}(s, m_{i,1})$, then $\text{dk} = \text{PRF}(s, (m, \text{msg}_r))$ is a valid one, but is still pseudo-random due to the freshness of msg_r generated by aDerR . Similarly, if $d_{r,2} = \text{PRF}(s, (m_{i,1}, m_{r,1}))$, then $\text{dk} = \text{PRF}(s, (\text{msg}_i, m))$ is a valid one, but is still pseudo-random due to the freshness of msg_i generated by alnit .

Clearly, the pseudo-randomness of valid dk holds even if the dictator additionally knows the randomness like $(d_{i,1}, d_{i,2}, d_{i,3})$ and $(d_{r,1}, d_{r,2}, d_{r,3})$. This yields strong PR-DK security.

1.3 Related Works

Anamorphic Cryptography. The notion of anamorphic encryption was proposed in [20]. Later works in [1, 5, 13, 14, 25] improved and extended this notion in different aspects. To be specific, more approaches to receiver-AME are provided in [1, 14]. The work in [1] decouples the generation of the anamorphic key-pair and the double key, and also proposes the notion of robustness for AME. Sender-AME was considered and specific constructions of robust sender-AME were presented in [25]. In [5], anamorphism is associated to homomorphic encryption, and the double key is dismantled with a public part and a secret part. In [13], anamorphism algorithms were extended to anamorphic signature.

Steganographic Key Exchange. Steganographic key exchange was firstly proposed in [23]. It aims to share a pseudo-random covert key by exchanging a sequence of seemingly normal messages. However, it only considered weak security where the adversary only implements passive attacks. Later, [11] proposed stronger requirement that permits the adversary to obtain the secret keys of parties. Nevertheless, steganographic key exchange does not allow active attacks in the security model, and hence much weaker than the security notions of AM-AKE defined in our paper.

2 Preliminary

Let $\kappa \in \mathbb{N}$ denote the security parameter and let pp denote the public parameter throughout the paper, and all algorithms, distributions, functions and adversaries take 1^κ and pp as implicit inputs. For $N \in \mathbb{N}$, define $[N] = \{1, 2, \dots, N\}$. If x is defined by y or the value of y is assigned to x , we write $x := y$. For a set \mathcal{X} , denote by $|\mathcal{X}|$ the number of elements in \mathcal{X} , and denote by $x \leftarrow_{\S} \mathcal{X}$ the procedure of sampling x from \mathcal{X} uniformly at random. If \mathcal{D} is distribution, $x \leftarrow_{\S} \mathcal{D}$ means that x is sampled according to \mathcal{D} . For an algorithm \mathcal{A} , let $y \leftarrow \mathcal{A}(x; r)$ or simply $y \leftarrow \mathcal{A}(x)$ denote running \mathcal{A} with input x and randomness r and assigning the output to y . ‘‘PPT’’ abbreviates probabilistic polynomial-time. Denote by poly

some polynomial function and negl some negligible function in κ . Let \perp denote the empty string/set, and all variables in our experiments are initialized to \perp .

Due to space limitations, we present the definitions of pseudo-random function (PRF), digital signature (SIG) and its EUF-CMA security, key encapsulation mechanism (KEM) and its IND-CPA security, two-pass authenticated key exchange (AKE) and the DDH assumption in the full version [24].

3 Anamorphic Authenticated Key Exchange

In this section, we present the syntax of anamorphic authenticated key exchange (AM-AKE), propose its robustness requirements, and define its security models. We also establish three impossibility results for plain AM-AKE, and define its relaxed security models.

3.1 Syntax of AM-AKE

Definition 1 ((Plain) Anamorphic Authenticated Key Exchange). *A two-pass authenticated key exchange scheme $\text{AKE} = (\text{Gen}, \text{Init}, \text{DerR}, \text{DerI})$ is called an AM-AKE scheme if there exists a corresponding anamorphic version of algorithms $(\text{aGen}, \text{alnit}, \text{aDerR}, \text{aDerI})$ with syntax defined below.*

- $(\text{apk}, \text{ask}, \text{aux}) \leftarrow \text{aGen}$: *The anamorphic key generation algorithm generates a pair of anamorphic public/secret keys (apk, ask) as well as an auxiliary message aux for storing extra secret information.*
- $(\text{amsg}_i, \text{st}, \text{aux}'_i) \leftarrow \text{alnit}(\text{apk}_r, \text{ask}_i, \text{aux}_i)$: *The anamorphic initialization algorithm takes an anamorphic public key apk_r of a responder (say P_r), an anamorphic secret key ask_i and an initiated auxiliary message aux_i of an initiator (say P_i) as input, and outputs a message amsg_i , a state st and an updated auxiliary message aux'_i for P_i .*
- $(\text{amsg}_r, K_r, \text{dk}_r) \leftarrow \text{aDerR}(\text{apk}_i, \text{ask}_r, \text{aux}_r, \text{amsg}_i)$: *The derivation algorithm for the responder takes an anamorphic public key apk_i of the initiator P_i , an anamorphic secret key ask_r and an initiated auxiliary message aux_r of the responder P_r , and a message amsg_i as input, and outputs a message amsg_r , a session key K_r and a double key dk_r for P_r .*
- $(K_i, \text{dk}_i) \leftarrow \text{aDerI}(\text{apk}_r, \text{ask}_i, \text{aux}'_i, \text{amsg}_r, \text{st})$: *The deterministic derivation algorithm for the initiator takes an anamorphic public key apk_r of the responder P_r , an anamorphic secret key ask_i and an updated auxiliary message aux'_i of the initiator P_i , a message amsg_r and a state st as input, and outputs a session key K_i and a double key dk_i for P_i .*

Then the AM-AKE scheme is denoted by $\text{AM-AKE} = ((\text{Gen}, \text{Init}, \text{DerR}, \text{DerI}), (\text{aGen}, \text{alnit}, \text{aDerR}, \text{aDerI}))$ where $(\text{Gen}, \text{Init}, \text{DerR}, \text{DerI})$ are called normal algorithms while $(\text{aGen}, \text{alnit}, \text{aDerR}, \text{aDerI})$ are called anamorphic algorithms.

If $\text{aux} = \perp$ or aux is generated independent of (apk, ask) in $(\text{apk}, \text{ask}, \text{aux}) \leftarrow \text{aGen}$, we call AM-AKE is a plain AM-AKE.

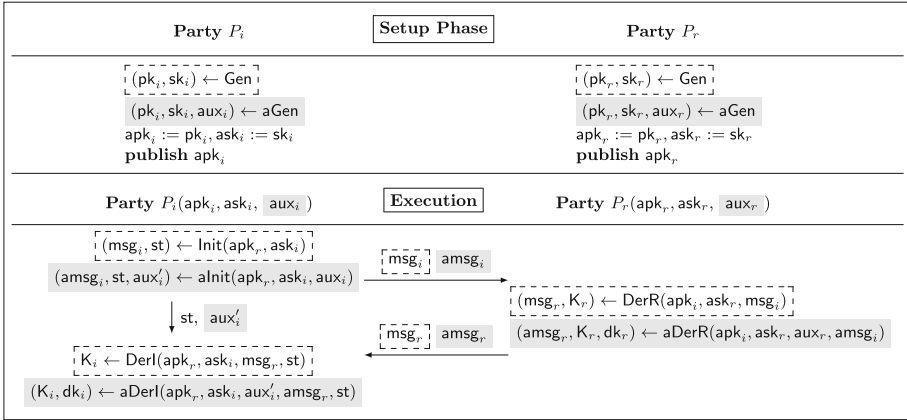


Fig. 1. The normal algorithms (with dotted boxes) and the anamorphic algorithms (with gray boxes) of AM-AKE.

An execution of an AM-AKE scheme AM-AKE is shown in Fig. 1. Any party can choose normal or anamorphic algorithms to run the AKE protocol, resulting in different working modes.

Working Modes of AM-AKE and Correctness Requirements. AM-AKE may work in the following three modes.

- **Normal Mode.** Both P_i and P_r invoke normal algorithms, i.e., executing the AKE protocol with (Init, DerR, Derl). But in the protocol execution, P_i may use either a normal key-pair (pk_i, sk_i) generated by Gen or an anamorphic key-pair (apk_i, ask_i) generated by aGen, and so does P_r .
- **Anamorphic Mode.** Both P_i and P_r invoke anamorphic algorithms, i.e., executing the protocol with (alnit, aDerR, aDerl), where both P_i and P_r have anamorphic keys (apk_i, ask_i, aux_i) , (apk_r, ask_r, aux_r) generated by aGen.
- **Half Mode.** One party invokes normal algorithms while the other invokes anamorphic algorithms. There are two cases described below.
 - **Case I.** P_i invokes anamorphic algorithms (alnit, aDerl) with its anamorphic keys (apk_i, ask_i, aux_i) , while P_r invokes normal algorithm DerR with either normal key-pair (pk_r, sk_r) or anamorphic key-pair (apk_r, ask_r) . In this case, P_i and P_r execute the protocol with (alnit, DerR, aDerl).
 - **Case II.** P_i invokes normal algorithms (Init, Derl) with either normal key-pair (pk_i, sk_i) or anamorphic key-pair (apk_i, ask_i) , while P_r invokes anamorphic algorithm aDerR with its anamorphic keys (apk_r, ask_r, aux_r) . In this case, P_i and P_r execute the protocol with (Init, aDerR, Derl).

For each of the above three working modes, P_i and P_r should derive the same session key $K_i = K_r$. Meanwhile, in the anamorphic mode, they should also derive the same double key $dk_i = dk_r$ besides the same session key.

Moreover, AM-AKE always considers adversaries(dictators) who has already obtained secret keys sk_i/ask_i and sk_r/ask_r from users, the state st from the initiator, and the derived session keys K_i, K_r from both initiator and responder. Therefore, an adversary can always invoke $Derl$ to obtain a session key K'_i . To avoid the detection of using anamorphic algorithms in AM-AKE, a basic requirement is that $Derl$ and $aDerl$ results in the same session key $K'_i = K_i$. These capture the correctness of AM-AKE.

Formally, we present the definition of correctness as follows, where different requirements serve for different working modes of AM-AKE. We also refer to Table 1 for a summary of correctness requirements in different working modes.

Definition 2 (Correctness of AM-AKE). *Let AM-AKE = ((Gen, Init, DerR, Derl), (aGen, alnit, aDerR, aDerl)) be an AM-AKE scheme. We consider the correctness for its three modes.*

Correctness for the normal mode. *If both P_i and P_r invoke normal algorithms in the AKE protocol, then it results in the same session key $K_i = K_r$, no matter P_i (and P_r) uses normal key-pair or anamorphic key-pair. More precisely, for any $(\overline{pk}_i, \overline{sk}_i) := (pk_i, sk_i)$ generated by Gen or $(\overline{pk}_i, \overline{sk}_i) := (apk_i, ask_i)$ generated by aGen, and for any $(\overline{pk}_r, \overline{sk}_r) := (pk_r, sk_r)$ generated by Gen or $(\overline{pk}_r, \overline{sk}_r) := (apk_r, ask_r)$ generated by aGen, we have*

$$\Pr \left[K_i = K_r \neq \perp \mid \begin{array}{l} (msg_i, st) \leftarrow \text{Init}(\overline{pk}_r, \overline{sk}_i) \\ (msg_r, K_r) \leftarrow \text{DerR}(\overline{pk}_i, \overline{sk}_r, msg_i) \\ K_i \leftarrow \text{Derl}(\overline{pk}_r, \overline{sk}_i, msg_r, st) \end{array} \right] = 1.$$

Correctness for the anamorphic mode. *If both P_i and P_r invoke anamorphic algorithms in the AKE protocol, then it results in the same session key $K_i = K_r$ and the same double key $dk_i = dk_r$. Meanwhile, the normal derivation by $Derl$ using ask_i should also result in the same session key $K'_i = K_i = K_r$. More precisely, for any $(apk_i, ask_i, aux_i) \leftarrow aGen$ and any $(apk_r, ask_r, aux_r) \leftarrow aGen$, we have*

$$\Pr \left[\begin{array}{l} K_i = K_r = K'_i \neq \perp \\ \wedge dk_i = dk_r \neq \perp \end{array} \mid \begin{array}{l} (amsg_i, st, aux'_i) \leftarrow \text{alnit}(apk_r, ask_i, aux_i) \\ (amsg_r, K_r, dk_r) \leftarrow \text{aDerR}(apk_i, ask_r, aux_r, amsg_i) \\ (K_i, dk_i) \leftarrow \text{aDerl}(apk_r, ask_i, aux'_i, amsg_r, st) \\ K'_i \leftarrow \text{Derl}(apk_r, ask_i, amsg_r, st) \end{array} \right] = 1.$$

Correctness for the half mode. *If one party invokes normal algorithms and the other invokes anamorphic algorithms, then the half mode still results in the same session key $K_i = K_r$. Moreover, the normal derivation $Derl$ using ask_i should also result in the same session key $K'_i = K_i = K_r$. More precisely, for any $(apk_i, ask_i, aux_i) \leftarrow aGen$, and for any $(\overline{pk}_r, \overline{sk}_r) := (pk_r, sk_r)$ generated by Gen or $(\overline{pk}_r, \overline{sk}_r) := (apk_r, ask_r)$ generated by aGen, we have*

$$\Pr \left[K_i = K_r = K'_i \neq \perp \mid \begin{array}{l} (amsg_i, st, aux'_i) \leftarrow \text{alnit}(\overline{pk}_r, ask_i, aux_i) \\ (msg_r, K_r) \leftarrow \text{DerR}(apk_i, \overline{sk}_r, amsg_i) \\ (K_i, dk_i) \leftarrow \text{aDerl}(\overline{pk}_r, ask_i, aux'_i, msg_r, st) \\ K'_i \leftarrow \text{Derl}(\overline{pk}_r, ask_i, msg_r, st) \end{array} \right] = 1.$$

On the other hand, for any $(\text{apk}_r, \text{ask}_r, \text{aux}_r) \leftarrow \text{aGen}$, and for any $(\overline{\text{pk}}_i, \overline{\text{sk}}_i) := (\text{pk}_i, \text{sk}_i)$ generated by Gen or $(\overline{\text{pk}}_i, \overline{\text{sk}}_i) := (\text{apk}_i, \text{ask}_i)$ generated by aGen, we have

$$\Pr \left[\text{K}_i = \text{K}_r \neq \perp \mid \begin{array}{l} (\text{msg}_i, \text{st}) \leftarrow \text{Init}(\text{apk}_r, \overline{\text{sk}}_i) \\ (\text{amsg}_r, \text{K}_r, \text{dk}_r) \leftarrow \text{aDerR}(\overline{\text{pk}}_i, \text{ask}_r, \text{aux}_r, \text{msg}_i) \\ \text{K}_i \leftarrow \text{Derl}(\text{apk}_r, \overline{\text{sk}}_i, \text{amsg}_r, \text{st}) \end{array} \right] = 1.$$

3.2 Robustness of AM-AKE

In practice, it is hard for P_i and P_r to agree on the working mode beforehand. So it happens AM-AKE works in *half mode*: one party invokes normal algorithms while the other invokes anamorphic algorithms. Accordingly, P_i and P_r can hardly agree on consistent double keys, so it is desirable for a party P invoking anamorphic algorithms to detect this issue and invalidate its double key by setting $\text{dk} = \perp$. This is captured by *robustness* of AM-AKE.

Roughly speaking, *robustness* of AM-AKE requires that in the half mode, except for the correctness of $\text{K}_i = \text{K}_r$, the party invoking anamorphic algorithms can detect the half mode of AKE and hence set its double key $\text{dk} := \perp$. According to whether the party is the initiator or the responder, we respectively define *initiator-robustness* and *responder-robustness* as follows.

Definition 3 (Initiator-Robustness). AM-AKE is called *initiator-robust*, if for any $(\text{apk}_i, \text{ask}_i, \text{aux}_i) \leftarrow \text{aGen}$, and for any $(\overline{\text{pk}}_r, \overline{\text{sk}}_r) := (\text{pk}_r, \text{sk}_r)$ generated by Gen or $(\overline{\text{pk}}_r, \overline{\text{sk}}_r) := (\text{apk}_r, \text{ask}_r)$ generated by aGen, we have

$$\Pr \left[\text{dk}_i = \perp \mid \begin{array}{l} (\text{amsg}_i, \text{st}, \text{aux}'_i) \leftarrow \text{alnit}(\overline{\text{pk}}_r, \text{ask}_i, \text{aux}_i) \\ (\text{msg}_r, \text{K}_r) \leftarrow \text{DerR}(\text{apk}_i, \overline{\text{sk}}_r, \text{amsg}_i) \\ (\text{K}_i, \text{dk}_i) \leftarrow \text{aDerl}(\overline{\text{pk}}_r, \text{ask}_i, \text{aux}'_i, \text{msg}_r, \text{st}) \end{array} \right] \geq 1 - \text{negl}(\kappa).$$

Definition 4 (Responder-Robustness). AM-AKE is called *responder-robust*, if for any $(\overline{\text{pk}}_r, \overline{\text{ask}}_r, \text{aux}_r) \leftarrow \text{aGen}$, and for any $(\overline{\text{pk}}_i, \overline{\text{sk}}_i) := (\text{pk}_i, \text{sk}_i)$ generated by Gen or $(\overline{\text{pk}}_i, \overline{\text{sk}}_i) := (\text{apk}_i, \text{ask}_i)$ generated by aGen, we have

$$\Pr \left[\text{dk}_r = \perp \mid \begin{array}{l} (\text{msg}_i, \text{st}) \leftarrow \text{Init}(\text{apk}_r, \overline{\text{sk}}_i) \\ (\text{amsg}_r, \text{K}_r, \text{dk}_r) \leftarrow \text{aDerR}(\overline{\text{pk}}_i, \text{ask}_r, \text{aux}_r, \text{msg}_i) \\ \text{K}_i \leftarrow \text{Derl}(\text{apk}_r, \overline{\text{sk}}_i, \text{amsg}_r, \text{st}) \end{array} \right] \geq 1 - \text{negl}(\kappa).$$

We stress that robustness is important for an AM-AKE scheme, because it's meaningless for a party to derive an un-agreed double key without the other party realizing it. Indeed, using un-agreed double key in the later anamorphic encryption/signature schemes has no effect at all.

For better illustration, we list all working modes of AM-AKE and the corresponding correctness and robustness requirements in Table 1.

Table 1. Working modes of AM-AKE and the corresponding correctness and robustness requirements. In column **Algorithms invoked by**, it indicates the type of algorithms invoked by P_i and P_r . In column **Correctness**, it shows the correctness requirements, where K_i and dk_i (resp., K_r and dk_r) denote the session key and double key derived by P_i (resp., P_r), and K'_i denotes the session key derived from DerI when P_i invokes anamorphic algorithms. In column **Robustness**, it shows the robustness requirements, where **Init-Rob./Resp-Rob.** denotes Initiator-Robustness/Responder-Robustness and “–” means no requirement.

Working Mode of AM-AKE	Algorithms invoked by		Correctness	Robustness	
	P_i	P_r		Init-Rob.	Resp-Rob.
Normal	Normal	Normal	$K_i = K_r$	–	–
Half	Normal	Anamorphic	$K_i = K_r$	–	$dk_r = \perp$
	Anamorphic	Normal	$K_i = K_r = K'_i$	$dk_i = \perp$	–
Anamorphic	Anamorphic	Anamorphic	$K_i = K_r = K'_i \wedge dk_i = dk_r$	–	–

3.3 Security Model for AM-AKE

In this subsection, we introduce the security models for AM-AKE. To this end, we need to capture the dictator(government)’s demands and behaviors to formalize the adversary. We consider the setting of multiple parties. In practice, the dictator may force every party involved in AM-AKE to surrender their secret keys, and reveal the session keys along with the state of the initiator in any completed AM-AKE session. Moreover, the dictator may impersonate any party and conduct active attacks because it owns the secret keys of all parties.

Intuitively, the security for AM-AKE requires that such a dictator cannot tell whether AM-AKE is working in the normal mode or in other modes. This is called Indistinguishability of Working Modes (IND-WM). Moreover, the double keys dk derived from the anamorphic mode will be used later by the anamorphic public-key primitives. To guarantee the security of the anamorphic public-key primitives, we have to require Pseudo-Randomness of Double Keys (PR-DK).

We also define the corresponding *strong* version of IND-WM and PR-DK by allowing the dictator additionally receive the internal randomness that all parties used in the seemingly benign AKE sessions, i.e., receiving the true randomness when normal algorithms are invoked while receiving simulated randomness when anamorphic algorithms are used. Especially, we require that there exists PPT simulator $\text{Sim} = (\text{SimI}, \text{SimR})$, where SimI can explain a randomness R'_i used by aInit as a randomness R_i of Init , and similarly, SimR can explain a randomness R'_r used by aDerR as a randomness R_r of DerR .¹ These result in strong IND-WM and strong PR-DK, denoted by sIND-WM and sPR-DK respectively.

More precisely, we define the formal security models with IND-WM/sIND-WM experiments $\text{Exp}_{\text{AM-AKE}, \mathcal{A}, N}^{\text{IND-WM}} / \text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{sIND-WM}}$ in Fig. 2 and PR-DK/sPR-DK experiments $\text{Exp}_{\text{AM-AKE}, \mathcal{A}, N}^{\text{PR-DK}} / \text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{sPR-DK}}$ in Fig. 3. To be clearer, we explain the local variables used in these security experiments.

¹ Note that the (anamorphic) derivation algorithms DerI and aDerI for the initiator are typically deterministic without using any randomness.

- sID : The identifier of a specific AKE session.
- $init[sID]$: The initiator of session sID .
- $resp[sID]$: The responder of session sID .
- $mode_I[sID]$: The working mode of the initiator in session sID .
- $M_I^{out}[sID]/M_I^{in}[sID]$: The message sent and received by the initiator in session sID .
- $M_R^{out}[sID]/M_R^{in}[sID]$: The message sent and received by the responder in session sID .
- $S[sID]$: The state of the initiator in session sID .
- $Aux[sID]$: The *updated* auxiliary message generated by the initiator in session sID .
- $K_I[sID]$ (resp., $K_R[sID]$): The session key generated by the initiator (resp., responder) in session sID .
- $DK[sID, P \in \{I, R\}]$: The double key generated by the initiator when $P = I$ or by the responder when $P = R$ in session sID .
- DK : The key space of double keys.
- $\mathcal{O}_{New}(i, r)$: The oracle establishes a new session for initiator P_i and responder P_r .
- $\mathcal{O}_{Init}(sID)$: The oracle invokes the initialization algorithm for session sID .
- $\mathcal{O}_{Der}(sID, m)$: The oracle invokes the derivation algorithm with input message m for the responder of session sID .
- $\mathcal{O}_{DerI}(sID, m)$: The oracle invokes the derivation algorithm with input message m for the initiator of session sID .
- $\mathcal{O}_{TestDK}(sID, P \in \{I, R\})$: The oracle provides either the double key generated by P of session sID or a random string. Note that this oracle can be invoked only once for each session to avoid trivial attack.

Especially, to formalize the IND-WM/sIND-WM security, we first require that the normal key-pair (pk, sk) generated by Gen and the anamorphic key-pair (apk, ask) generated by $aGen$ are computationally indistinguishable, and then we can choose the keys of all parties via $aGen$. During the experiments (cf. Fig. 2), the adversary is allowed to designate the working modes of the initiator and the responder by providing additionally variables $w_I, w_R \in \{\mathbf{N}, \mathbf{A}\}$ to oracles \mathcal{O}_{Init} and \mathcal{O}_{DerR} , respectively. The adversary is asked to tell whether the oracles run the protocols in the normal modes or in the modes specified by the adversary.

As for the PR-DK/sPR-DK security (cf. Fig. 3), all parties work in the anamorphic modes, and the adversary is asked to distinguish real double keys dk from uniformly chosen keys via a \mathcal{O}_{TestDK} oracle. To avoid trivial attacks, we define the notion of matching sessions as follows, and we require that the adversary cannot test the double keys of matching sessions.

Definition 5 (Matching Sessions). *For two sessions sID, sID^* and two parties $P, \bar{P} \in \{I, R\}$, we say (sID, P) and (sID^*, \bar{P}) match, if the same parties are involved (i.e., $init[sID], resp[sID] = (init[sID^*], resp[sID^*])$), the messages sent and received are the same (i.e., $(M_P^{in}[sID], M_P^{out}[sID]) = (M_{\bar{P}}^{out}[sID^*], M_{\bar{P}}^{in}[sID^*])$), and the parties are of different type (i.e., $\bar{P} = \{I, R\} \setminus P$). In particular, we define*

$$\mathfrak{M}[sID, P] := \left\{ (sID^*, \bar{P}) \mid \begin{array}{l} (init[sID], resp[sID]) = (init[sID^*], resp[sID^*]) \wedge \bar{P} = \{I, R\} \setminus P \\ \wedge (M_P^{in}[sID], M_P^{out}[sID]) = (M_{\bar{P}}^{out}[sID^*], M_{\bar{P}}^{in}[sID^*]) \end{array} \right\}$$

as the set of matching sessions with (sID, P) .

Now we are ready to present the formal definition of the security of AM-AKE.

Definition 6 (Security of AM-AKE). *The security of AM-AKE contains indistinguishability of working modes (IND-WM) and pseudo-randomness of double keys (PR-DK).*

<pre> Exp_{AM-AKE, A, N}^{IND-WM} / Exp_{AM-AKE, A, Sim, N}^{sIND-WM} / Exp_{AM-AKE, A, Sim, N}^{relaxed-IND-WM} / Exp_{AM-AKE, A, Sim, N}^{relaxed-sIND-WM} b ←_S {0, 1} cnt := 0 //session counter for n ∈ [N]: (apk_n, ask_n, aux_n) ← aGen b' ← A^{OWM}(apk₁, ..., apk_N, ask₁, ..., ask_N) return b' = b O_{New}(i, r): if i ∉ [N] or r ∉ [N] or i = r: return ⊥ cnt++ sID := cnt init[sID] := i; resp[sID] := r return sID O_{Init}(sID, w₁ ∈ {N, A}): if init[sID] = ⊥: return ⊥ //session not established if M_i^{out}[sID] ≠ ⊥: return ⊥ //no re-use (i, r) := (init[sID], resp[sID]) if b = 0: //normal working mode (msg_i, st) ← Init(apk_r, ask_i; R_i) if b = 1: //working mode specified by A if w₁ = N: (msg_i, st) ← Init(apk_r, ask_i; R_i) if w₁ = A: (msg_i, st, aux'_i) ← anit(apk_r, ask_i, aux_i; R'_i) Aux[sID] := aux'_i R_i ← Siml(apk_r, ask_i, aux_i, R'_i) mode_i[sID] := w₁ M_i^{out}[sID] := msg_i S[sID] := st return (msg_i, st, R_i) </pre>	<pre> O_{DerR}(sID, m, w_R ∈ {N, A}): if M_i^{out}[sID] = ⊥: return ⊥ //initiator not invoked if M_R^{out}[sID] ≠ ⊥: return ⊥ //no re-use if m ≠ M_i^{out}[sID] ∧ w_R = A: return ⊥ //active attack (i, r) := (init[sID], resp[sID]) if b = 0: //normal working mode (msg_r, K_r) ← DerR(apk_i, ask_r, m; R_r) if b = 1: //working mode specified by A if w_R = N: (msg_r, K_r) ← DerR(apk_i, ask_r, m; R_r) if w_R = A: (msg_r, K_r, dk_r) ← aDerR(apk_i, ask_r, aux_r, m; R'_r) R_r ← SimR(apk_i, ask_r, aux_r, m, R'_r) K_R[sID] := K_r M_R^{out}[sID] := msg_r return (msg_r, K_r, R_r) O_{DerI}(sID, m): if M_R^{out}[sID] = ⊥: return ⊥ //responder not invoked if K_i[sID] ≠ ⊥: return ⊥ //no re-use (i, r) := (init[sID], resp[sID]) w₁ := mode_i[sID] st := S[sID] if b = 0: //normal working mode K_i ← DerI(apk_r, ask_i, m, st) if b = 1: //working mode specified by A if w₁ = N: K_i ← DerI(apk_r, ask_i, m, st) if w₁ = A: aux'_i := Aux[sID] (K_i, dk_i) ← aDerI(apk_r, ask_i, aux'_i, m, st) K_I[sID] := K_i return K_i </pre>
---	---

Fig. 2. Security experiments for defining IND-WM (without gray and dotted boxes) and **sIND-WM** (with gray boxes) of AM-AKE, and experiments for defining **relaxed IND-WM** (with dotted boxes) and **relaxed sIND-WM** (with both gray and dotted boxes) of plain AM-AKE, where $\mathcal{O}_{\text{WM}} := \{\mathcal{O}_{\text{New}}, \mathcal{O}_{\text{Init}}, \mathcal{O}_{\text{DerR}}, \mathcal{O}_{\text{DerI}}\}$. Here R_i , R'_i , R_r and R'_r are uniformly sampled from the corresponding randomness spaces.

– **Indistinguishability of Working Modes (IND-WM).** For any PPT adversary \mathcal{A} and any $N = \text{poly}(\kappa)$, it holds that $|\Pr[\mathcal{A}(\text{pk}, \text{sk}) = 1] - \Pr[\mathcal{A}(\text{apk}, \text{ask}) = 1]| \leq \text{negl}(\kappa)$, where $(\text{pk}, \text{sk}) \leftarrow \text{Gen}$ and $(\text{apk}, \text{ask}, \text{aux}) \leftarrow \text{aGen}$, and

$$\left| \Pr \left[\text{Exp}_{\text{AM-AKE}, \mathcal{A}, N}^{\text{IND-WM}} = 1 \right] - \frac{1}{2} \right| \leq \text{negl}(\kappa). \quad (1)$$

– **Pseudo-Randomness of Double Keys (PR-DK).** For any PPT adversary \mathcal{A} and any $N = \text{poly}(\kappa)$, it holds that

$$\left| \Pr \left[\text{Exp}_{\text{AM-AKE}, \mathcal{A}, N}^{\text{PR-DK}} = 1 \right] - \frac{1}{2} \right| \leq \text{negl}(\kappa). \quad (2)$$

The strong security of AM-AKE includes strong indistinguishability of working modes (sIND-WM) and strong pseudo-randomness of double keys (sPR-DK),

$\text{Exp}_{\text{AM-AKE}, \mathcal{A}, N}^{\text{PR-DK}} / \text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{sPR-DK}} /$ <div style="border: 1px dashed black; padding: 2px; margin: 2px;"> $\text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{relaxed-PR-DK}} /$ </div> <div style="border: 1px dashed black; padding: 2px; margin: 2px;"> $\text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{relaxed-sPR-DK}} /$ </div> <p> $b \leftarrow_{\mathcal{S}} \{0, 1\}$ $\text{cnt} := 0$ //session counter for $n \in [N]$: $(\text{apk}_n, \text{ask}_n, \text{aux}_n) \leftarrow \text{aGen}$ $b' \leftarrow \mathcal{A}^{\text{OPRD}}(\text{apk}_1, \dots, \text{apk}_N, \text{ask}_1, \dots, \text{ask}_N)$ return $b' = b$ </p> <p> $\mathcal{O}_{\text{New}}(i, r)$: if $i \notin [N]$ or $r \notin [N]$ or $i = r$: return \perp $\text{cnt}++$ $\text{sID} := \text{cnt}$ $\text{init}[\text{sID}] := i$ $\text{resp}[\text{sID}] := r$ return sID </p> <p> $\mathcal{O}_{\text{Init}}(\text{sID})$: if $\text{init}[\text{sID}] = \perp$: //session not established return \perp if $M_i^{\text{out}}[\text{sID}] \neq \perp$: //no re-use return \perp $(i, r) := (\text{init}[\text{sID}], \text{resp}[\text{sID}])$ $(\text{ams}_{g_i}, \text{st}, \text{aux}'_i) \leftarrow \text{alnit}(\text{apk}_r, \text{ask}_i, \text{aux}_i; R'_i)$ $R_i \leftarrow \text{Siml}(\text{apk}_r, \text{ask}_i, \text{aux}_i, R'_i)$ $M_i^{\text{out}}[\text{sID}] := \text{ams}_{g_i}$ $S[\text{sID}] := \text{st}$ $\text{Aux}[\text{sID}] := \text{aux}'_i$ return $(\text{ams}_{g_i}, \text{st}, R_i)$ </p>	$\mathcal{O}_{\text{DerR}}(\text{sID}, m)$: if $M_i^{\text{out}}[\text{sID}] = \perp$: return \perp //initiator not invoked if $M_R^{\text{out}}[\text{sID}] \neq \perp$: return \perp //no re-use <div style="border: 1px dashed black; padding: 2px;"> if $m \neq M_i^{\text{out}}[\text{sID}]$: return \perp //active attack </div> $(i, r) := (\text{init}[\text{sID}], \text{resp}[\text{sID}])$ $(\text{ams}_{g_r}, K_r, \text{dk}_r) \leftarrow \text{aDerR}(\text{apk}_i, \text{ask}_r, \text{aux}_r, m; R'_r)$ $R_r \leftarrow \text{SimR}(\text{apk}_r, \text{ask}_r, \text{aux}_r, m, R'_r)$ $M_R^{\text{in}}[\text{sID}] := m; M_R^{\text{out}}[\text{sID}] := \text{ams}_{g_r}$ $\text{DK}[\text{sID}, R] := \text{dk}_r$ return $(\text{ams}_{g_r}, K_r, R_r)$ <p> $\mathcal{O}_{\text{DerI}}(\text{sID}, m)$: if $M_R^{\text{out}}[\text{sID}] = \perp$: return \perp //responder not invoked if $K_i[\text{sID}] \neq \perp$: return \perp //no re-use <div style="border: 1px dashed black; padding: 2px;"> if $m \neq M_R^{\text{out}}[\text{sID}]$: return \perp //active attack </div> $(i, r) := (\text{init}[\text{sID}], \text{resp}[\text{sID}])$ $\text{st} := S[\text{sID}]$ $\text{aux}'_i := \text{Aux}[\text{sID}]$ $(K_i, \text{dk}_i) \leftarrow \text{aDerI}(\text{apk}_r, \text{ask}_i, \text{aux}'_i, m, \text{st})$ $M_i^{\text{in}}[\text{sID}] := m$ $\text{DK}[\text{sID}, I] := \text{dk}_i$ return K_i </p> <p> $\mathcal{O}_{\text{TestDK}}(\text{sID}, P \in \{I, R\})$: if $\text{DK}[\text{sID}, P] = \perp$: return \perp //dk not generated or invalid if $\exists (\text{sID}^*, \bar{P}) \in \mathfrak{M}[\text{sID}, P]$ and $\text{test}[\text{sID}^*, \bar{P}] = 1$ return \perp //trivial attack $\text{test}[\text{sID}, P] := 1$ if $b = 1$: return $\text{DK}[\text{sID}, P]$ else : return $d \leftarrow_{\mathcal{S}} \text{DK}$ </p>
---	---

Fig. 3. Security experiments for defining PR-DK (without gray and dotted boxes) and sPR-DK (with gray boxes) of AM-AKE, and experiments for defining relaxed PR-DK (with dotted boxes) and relaxed sPR-DK (with both gray and dotted boxes) of plain AM-AKE, where $\mathcal{O}_{\text{PRD}} := \{\mathcal{O}_{\text{New}}, \mathcal{O}_{\text{Init}}, \mathcal{O}_{\text{DerR}}, \mathcal{O}_{\text{DerI}}, \mathcal{O}_{\text{TestDK}}\}$. Here R'_i and R'_r are uniformly sampled from the corresponding randomness spaces.

which require that there exists PPT simulator $\text{Sim} = (\text{Siml}, \text{SimR})$ such that the above (1) and (2) hold for $\text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{sIND-WM}}$ and $\text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{sPR-DK}}$ experiments.

3.4 Impossibility Results and Relaxed Security for Plain AM-AKE

In this subsection, we show three impossibility results for (two-pass) plain AM-AKE, and then define proper *relaxed* security to circumvent the impossibility results. More precisely, we show the impossibility results via the following three theorems, whose formal proofs are shown in the full version [24], and we refer to Subsect. 1.2 for a high-level overview of the proofs. Roughly speaking, for a (two-pass) plain AM-AKE, the adversary \mathcal{A} holds both $(\text{apk}_i, \text{ask}_i)$ and $(\text{apk}_r, \text{ask}_r)$, and thus a null $\text{aux} = \perp$ or independent aux does not offer any advantage to P_i or P_r over \mathcal{A} , and \mathcal{A} is capable of doing whatever P_i or P_r can do.

Theorem 1. *It is impossible for a two-pass plain AM-AKE scheme AM-AKE to achieve responder-robustness.*

Theorem 2. *If a plain AM-AKE scheme AM-AKE is initiator-robust, then it is impossible for AM-AKE to achieve the IND-WM/sIND-WM security.*

Theorem 3. *It is impossible for a plain AM-AKE scheme AM-AKE to achieve the PR-DK/sPR-DK security.*

To circumvent the above impossibility results for plain AM-AKE, we weaken the IND-WM/sIND-WM security and PR-DK/sPR-DK security, by restricting the active attacks by adversary. More precisely, we disallow the adversary to query $\mathcal{O}_{\text{DerR}}(\text{sID}, m, w_R = \mathbf{A})$ with its own messages m when $w_R = \mathbf{A}$ in the IND-WM/sIND-WM experiments, and disallow the adversary to query $\mathcal{O}_{\text{DerR}}(\text{sID}, m)$ and $\mathcal{O}_{\text{DerI}}(\text{sID}, m)$ with its own messages m in the PR-DK/sPR-DK experiments, respectively. These yield *relaxed* IND-WM/sIND-WM and *relaxed* PR-DK/sPR-DK securities, with experiments shown in Fig. 2 and Fig. 3 with [dashed boxes].

Definition 7 (Relaxed Security of Plain AM-AKE). *The relaxed security of plain AM-AKE contains relaxed IND-WM/sIND-WM and relaxed PR-DK/sPR-DK, which are defined the same as those (non-relaxed versions) of AM-AKE in Definition 6, except that the experiments are replaced by $\text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{relaxed-IND-WM}} / \text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{relaxed-sIND-WM}} / \text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{relaxed-PR-DK}} / \text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{relaxed-sPR-DK}}$ in Fig. 2 and Fig. 3, respectively.*

In the full version [24], we present a generic construction of plain AM-AKE with relaxed security, which not only achieves relaxedsIND-WM and relaxedsPR-DK security, but also enjoys initiator-robustness. We also discuss how to achieve responder-robustness by relying on more passes to evade the first impossibility result. (See Subsect. 1.2 for a high-level overview of this plain AM-AKE construction and its security analysis.) Then we show how to instantiate the generic construction from the popular SIG+KEM and three-KEM paradigms for constructing AKE and get the corresponding plain AM-AKE schemes.

4 Generic Construction of Robust & Strongly-Secure AM-AKE from AKE

In this section, we present a generic construction of robust and strongly-secure AM-AKE from a basic AKE with the help of a PRF. To make the construction possible, the underlying AKE should be equipped with some new properties, which are defined in Subsect. 4.1. We call such AKE as *qualified AKE*. Then we show the generic construction in Subsect. 4.2 and present its security proof in Subsect. 4.3.

4.1 New Properties for Functions and Algorithms

To characterize the conditions on the basic AKE scheme, in this subsection, we first define three new properties for general functions and algorithms. Roughly speaking, the *entropy-preserving* property of a function asks the function output to have negligible guessing probability on uniformly random input. The *η -separable* property of an algorithm means that the first $\eta - 1$ parts of the output can be computed publicly and in a way independent of the input. The *secret extractability of a key generation algorithm* Gen requires that the key-pair (pk, sk) from Gen can be perfectly simulated by an algorithm SimGen which additionally outputs a master key msk, and it enables the extraction of a pseudo-random secret s from msk and pk' of another party via an algorithm Extract.

Definition 8 (Entropy-Preserving Function). A function $f : \mathcal{X} \rightarrow \mathcal{Y}$ is entropy-preserving, if for any $y \in \mathcal{Y}$, it holds $\Pr[f(x) = y | x \leftarrow_{\S} \mathcal{X}] \leq \text{negl}(\kappa)$.

Definition 9 (η -Separable Algorithm). Let $\eta \in \mathbb{N}$, and let $(y, z) \leftarrow \text{Alg}(x)$ be a PPT algorithm which inputs x and outputs (y, z) . We say that Alg is η -separable for generating y if Alg can be implemented with $(f_1, \dots, f_{\eta-1}, \overline{\text{Alg}})$ as follows, where $f_j : \mathcal{D}_j \rightarrow \{0, 1\}^*$ is a publicly and efficiently computable function for $j \in [\eta - 1]$, and $\overline{\text{Alg}}$ is a PPT algorithm.

- $(y, z) \leftarrow \text{Alg}(x)$: For $j \in [\eta - 1]$, sample $d_j \leftarrow_{\S} \mathcal{D}_j$ and compute $m_j := f_j(d_j)$; invoke $(m_\eta, z) \leftarrow \overline{\text{Alg}}(x, d_1, \dots, d_{\eta-1})$; output $y := (m_1, \dots, m_\eta)$ and z .

Definition 10 (Secret Extractability of Gen). Let Gen be a key generation algorithm that outputs (pk, sk) .² We say Gen supports secret extractability if there exist two PPT algorithms SimGen and Extract satisfying the following properties.

- $(pk, sk, msk) \leftarrow \text{SimGen}$: it is a simulated key generation algorithm that outputs a simulated key-pair (pk, sk) together with a master key msk.
- $s \leftarrow \text{Extract}(msk_i, pk_r)$: it is a deterministic extracting algorithm that takes a master key msk_i and a public key pk_r as input, and outputs a secret $s \in \mathcal{D}_E$.

Identically Distributed Key-Pairs. The simulated key-pair has the same distribution as the normal pair, i.e., the following two distributions are identical:

$$\{(pk, sk) \mid (pk, sk) \leftarrow \text{Gen}\} \equiv \{(pk, sk) \mid (pk, sk, msk) \leftarrow \text{SimGen}\}.$$

Extracting Correctness. For any $(pk_i, sk_i, msk_i) \leftarrow \text{SimGen}$ and $(pk_r, sk_r, msk_r) \leftarrow \text{SimGen}$, it holds that $\text{Extract}(msk_i, pk_r) = \text{Extract}(msk_r, pk_i)$.

Pseudo-Randomness of the Extracting. For any PPT adversary \mathcal{A} , we have

$$\text{Adv}_{\text{Gen}, \mathcal{A}}^{\text{PR-Ext}}(\kappa) := |\Pr[\mathcal{A}(pk_i, pk_r, sk_i, sk_r, s_0) = 1] - \Pr[\mathcal{A}(pk_i, pk_r, sk_i, sk_r, s_1) = 1]| \leq \text{negl}(\kappa),$$

where $(pk_i, sk_i, msk_i) \leftarrow \text{SimGen}$, $(pk_r, sk_r, msk_r) \leftarrow \text{SimGen}$, $s_0 := \text{Extract}(msk_i, pk_r)$, and $s_1 \leftarrow_{\S} \mathcal{D}_E$.

Based on the three new properties, we are ready to describe the requirements on the basic AKE and present the generic construction of AM-AKE from it.

² Gen can be the key generation algorithm of any public-key primitive, like AKE, SIG, KEM, etc.

4.2 Construction of AM-AKE from AKE and PRF

Let $\text{AKE} = (\text{Gen}, \text{Init}, \text{DerR}, \text{DerI})$ be a two-pass AKE scheme that satisfies:

- Gen has *secret extractability*, supported by algorithms $(\text{SimGen}, \text{Extract})$ and secret space \mathcal{D}_E as per Definition 10;
- Init is *3-separable for generating msg_i* , supported by $(f_{1,1}, f_{1,2}, \overline{\text{Init}})$ as per Definition 9, i.e., $\text{Init}(\text{pk}_r, \text{sk}_i)$ generates $(\text{msg}_i, \text{st})$ by sampling $d_{i,1} \leftarrow_{\$} \mathcal{D}_{1,1}$, $d_{i,2} \leftarrow_{\$} \mathcal{D}_{1,2}$, computing $m_{i,1} := f_{1,1}(d_{i,1})$, $m_{i,2} := f_{1,2}(d_{i,2})$, invoking $(m_{i,3}, \text{st}) \leftarrow \overline{\text{Init}}(\text{pk}_r, \text{sk}_i, d_{i,1}, d_{i,2})$, and setting $\text{msg}_i := (m_{i,1}, m_{i,2}, m_{i,3})$;
- DerR is *3-separable for generating msg_r* , supported by $(f_{R,1}, f_{R,2}, \overline{\text{DerR}})$ as per Definition 9, i.e., $\text{DerR}(\text{pk}_i, \text{sk}_r, \text{msg}_i)$ generates (msg_r, K_r) by sampling $d_{r,1} \leftarrow_{\$} \mathcal{D}_{R,1}, d_{r,2} \leftarrow_{\$} \mathcal{D}_{R,2}$, computing $m_{r,1} := f_{R,1}(d_{r,1})$, $m_{r,2} := f_{R,2}(d_{r,2})$, invoking $(m_{r,3}, K_r) \leftarrow \overline{\text{DerR}}(\text{pk}_i, \text{sk}_r, \text{msg}_i, d_{r,1}, d_{r,2})$, and setting $\text{msg}_r := (m_{r,1}, m_{r,2}, m_{r,3})$;
- The functions $f_{1,1}, f_{1,2}, f_{R,1}, f_{R,2}$ associated with Init and DerR are *entropy-preserving* as per Definition 8.

We call such AKE as *qualified AKE*, with requirements summarized in Table 2. Moreover, let $\text{PRF} : \mathcal{D}_E \times \{0, 1\}^* \rightarrow \mathcal{D}_{1,2} \times \mathcal{D}_{R,2} \times \{0, 1\}^\kappa$ be a pseudo-random function. For ease of exposition, we parse the output of PRF as three parts, i.e., $\text{PRF}_I/\text{PRF}_R/\text{PRF}_D : \mathcal{D}_E \times \{0, 1\}^* \rightarrow \mathcal{D}_{1,2}/\mathcal{D}_{R,2}/\{0, 1\}^\kappa$, such that $\text{PRF}(s, m) = (\text{PRF}_I(s, m), \text{PRF}_R(s, m), \text{PRF}_D(s, m))$ for all $s \in \mathcal{D}_E, m \in \{0, 1\}^*$.

Table 2. Requirements for $\text{AKE} = (\text{Gen}, \text{Init}, \text{DerR}, \text{DerI})$ to be *qualified* for constructing AM-AKE.

Qualified AKE	Gen	Init	DerR
Requirements	<i>secret extractability</i>	<i>3-separable for msg_i with entropy-preserving $f_{1,1}, f_{1,2}$</i>	<i>3-separable for msg_r with entropy-preserving $f_{R,1}, f_{R,2}$</i>
Supportive Func./Alg.	$(\text{SimGen}, \text{Extract})$	$(f_{1,1}, f_{1,2}, \overline{\text{Init}})$	$(f_{R,1}, f_{R,2}, \overline{\text{DerR}})$

Now we convert AKE to an AM-AKE scheme $\text{AM-AKE} = ((\text{Gen}, \text{Init}, \text{DerR}, \text{DerI}), (\text{aGen}, \text{aInit}, \text{aDerR}, \text{aDerI}))$ with the help of PRF, where the anamorphic algorithms are described below. (See also Fig. 4 for an illustration of AM-AKE.)

- $(\text{apk}, \text{ask}, \text{aux}) \leftarrow \text{aGen}$: it invokes the simulated key generation algorithm $(\text{pk}, \text{sk}, \text{msk}) \leftarrow \overline{\text{SimGen}}$, and sets $(\text{apk}, \text{ask}) := (\text{pk}, \text{sk})$ and $\text{aux} := \text{msk}$.
- $(\text{amsg}_i, \text{st}, \text{aux}'_i) \leftarrow \text{aInit}(\text{apk}_r, \text{ask}_i, \text{aux}_i = \text{msk}_i)$: it first extracts a secret $s_i := \text{Extract}(\text{msk}_i, \text{apk}_r)$. Next it randomly chooses $d_{i,1} \leftarrow_{\$} \mathcal{D}_{1,1}$ and computes $m_{i,1} := f_{1,1}(d_{i,1})$. Then it computes $d_{i,2} := \text{PRF}_I(s_i, m_{i,1}) \in \mathcal{D}_{1,2}$, $m_{i,2} := f_{1,2}(d_{i,2})$, and invokes $(m_{i,3}, \text{st}) \leftarrow \overline{\text{Init}}(\text{apk}_r, \text{ask}_i, d_{i,1}, d_{i,2})$. Finally, it returns $(\text{amsg}_i := (m_{i,1}, m_{i,2}, m_{i,3}), \text{st}, \text{aux}'_i := (s_i, \text{amsg}_i))$.

- $(\text{amsgr}_r, K_r, \text{dk}_r) \leftarrow \text{aDerR}(\text{apk}_i, \text{ask}_r, \text{aux}_r = \text{msk}_r, \text{amsg}_i = (m_{i,1}, m_{i,2}, m_{i,3}))$: it first randomly chooses $d_{r,1} \leftarrow_{\$} \mathcal{D}_{R,1}$ and computes $m_{r,1} := f_{R,1}(d_{r,1})$. Next it extracts a secret $s_r := \text{Extract}(\text{msk}_r, \text{apk}_i)$, and computes $d_{r,2} := \text{PRF}_R(s_r, (m_{i,1}, m_{r,1})) \in \mathcal{D}_{R,2}$, $m_{r,2} := f_{R,2}(d_{r,2})$, invokes $(m_{r,3}, K_r) \leftarrow \text{DerR}(\text{apk}_i, \text{ask}_r, \text{amsg}_i, d_{r,1}, d_{r,2})$, and sets $\text{amsgr}_r := (m_{r,1}, m_{r,2}, m_{r,3})$. Afterwards, it checks whether $m_{i,2} = f_{i,2}(\text{PRF}_I(s_r, m_{i,1}))$ holds. If the check passes, then it sets $\text{dk}_r := \text{PRF}_D(s_r, (\text{amsg}_i, \text{amsgr}_r)) \in \{0, 1\}^\kappa$ as the double key; otherwise, it sets $\text{dk}_r := \perp$. Finally, it returns $(\text{amsgr}_r, K_r, \text{dk}_r)$.
- $(K_i, \text{dk}_i) \leftarrow \text{aDerI}(\text{apk}_r, \text{ask}_i, \text{aux}'_i = (s_i, \text{amsg}_i), \text{amsgr}_r = (m_{r,1}, m_{r,2}, m_{r,3}), \text{st})$: it first checks whether $m_{r,2} = f_{R,2}(\text{PRF}_R(s_i, (m_{i,1}, m_{r,1})))$ holds. If yes, it sets $\text{dk}_i := \text{PRF}_D(s_i, (\text{amsg}_i, \text{amsgr}_r)) \in \{0, 1\}^\kappa$ as the double key; else, $\text{dk}_i := \perp$. Finally, it invokes $K_i \leftarrow \text{DerI}(\text{apk}_r, \text{ask}_i, \text{amsgr}_r, \text{st})$, and returns (K_i, dk_i) .

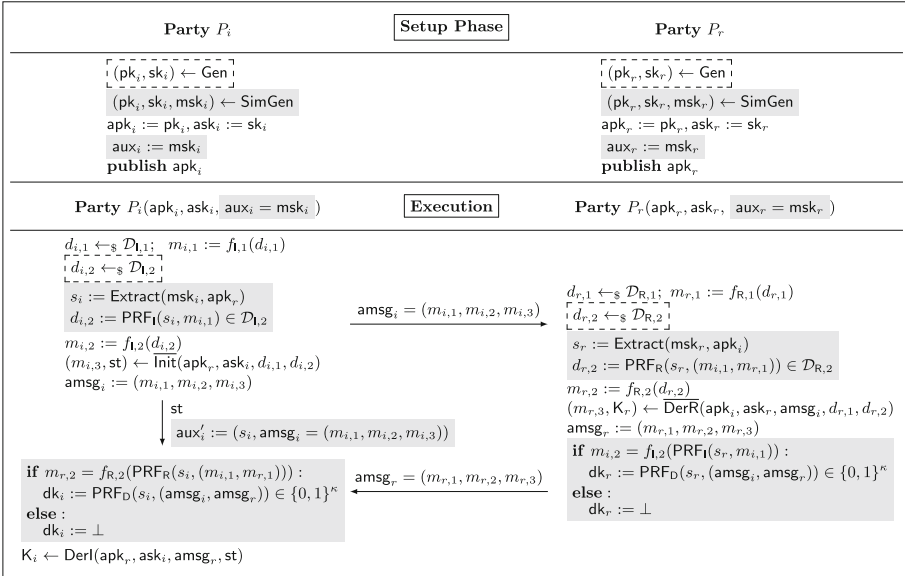


Fig. 4. Generic construction of the AM-AKE scheme AM-AKE based on AKE and PRF, where [dotted boxes] appear only in normal algorithms (Gen, Init, DerR, DerI), and [gray boxes] appear only in anamorphic algorithms (aGen, alnit, aDerR, aDerI).

Let us compare the normal algorithms and the anamorphic ones.

- The anamorphic algorithm aGen invokes SimGen to produce a simulated key-pair $(\text{apk}, \text{ask}) := (\text{pk}, \text{sk})$ as well as a master secret $\text{aux} := \text{msk}$. By the property of secret extractability of the normal algorithm Gen, the anamorphic key-pair has the same distribution as the normal key-pair generated by Gen.

- The normal algorithm `Init` makes use of random coins $d_{i,1}$ and $d_{i,2}$ for the generation of msg_i . The anamorphic algorithm `alnit` can be regarded as the normal `Init` taking random coins $d_{i,1}$ and specific coins $d_{i,2} = \text{PRF}_1(s_i, m_{i,1})$, with s_i a secret extracted from the master secret msk_i of P_i and apk_r of P_r .
- The normal algorithm `DerR` makes use of random coins $d_{r,1}, d_{r,2}$ for the generation of msg_r and the session key K_r . The anamorphic algorithm `aDerR` has two parts: one part can be regarded as the normal `DerR` taking random coins $d_{r,1}$ and specific coins $d_{r,2} = \text{PRF}_R(s_r, (m_{i,1}, m_{r,1}))$ to output msg_r and key K_r ; the other part is in charge of generating the double key $\text{dk}_r := \text{PRF}_D(s_r, (\text{amsg}_i, \text{amsg}_r))$ or $\text{dk}_r := \perp$ depending on whether $m_{i,2} = f_{1,2}(\text{PRF}_1(s_r, m_{i,1}))$ holds, with s_r a secret derived from the master secret msk_r of P_r and apk_i of P_i .
- The normal algorithm `DerI` is deterministic and outputs the session key K_i . The anamorphic algorithm `aDerI` functions identically as `DerI` for the generation of key K_i , but it is also in charge of generating the double key $\text{dk}_i := \text{PRF}_D(s_i, (\text{amsg}_i, \text{amsg}_r))$ or $\text{dk}_i := \perp$ depending on whether $m_{r,2} = f_{R,2}(\text{PRF}_R(s_i, (m_{i,1}, m_{r,1})))$ holds.

Note that the correctness of the underlying AKE guarantees that $K_i = K_r$ for every possible choices of $d_{i,1}, d_{i,2}, d_{r,1}, d_{r,2}$. Thus even using specific coins in the anamorphic algorithms, we also have $K_i = K_r$. This shows the correctness of $K_i = K_r$ in all working modes. Moreover, in the anamorphic mode, we have $\text{dk}_i = \text{PRF}_D(s_i, (\text{amsg}_i, \text{amsg}_r)) = \text{PRF}_D(s_r, (\text{amsg}_i, \text{amsg}_r)) = \text{dk}_r$ since $s_i = \text{Extract}(\text{msk}_i, \text{apk}_r) = \text{Extract}(\text{msk}_r, \text{apk}_i) = s_r$ holds by the extracting correctness of `Gen`'s secret extractability, and thus the correctness of double key holds.

Below we analyze the robustness of our AM-AKE.

Initiator-Robustness. Suppose that P_i invokes anamorphic algorithms `alnit` and `aDerI` while P_r invokes normal algorithm `DerR`, then P_r computes $m_{r,2} := f_{R,2}(d_{r,2})$ by using a uniformly chosen $d_{r,2} \leftarrow_{\S} \mathcal{D}_{R,2}$. When P_i invokes the anamorphic algorithm `aDerI` to check if $m_{r,2} = f_{R,2}(\text{PRF}_R(s_i, (m_{i,1}, m_{r,1})))$ holds, we know that $f_{R,2}(\text{PRF}_R(s_i, (m_{i,1}, m_{r,1})))$ is independent of $m_{r,2} := f_{R,2}(d_{r,2})$ since $d_{r,2} \leftarrow_{\S} \mathcal{D}_{R,2}$ is chosen independently of $s_i, m_{i,1}, m_{r,1}$ by P_r . Thus for every possible value of $f_{R,2}(\text{PRF}_R(s_i, (m_{i,1}, m_{r,1})))$, the check $m_{r,2} := f_{R,2}(d_{r,2}) = f_{R,2}(\text{PRF}_R(s_i, (m_{i,1}, m_{r,1})))$ can pass with only a negligible probability by the entropy-preserving property of $f_{R,2}$ and due to the randomness of $d_{r,2} \leftarrow_{\S} \mathcal{D}_{R,2}$, and consequently, P_i will set $\text{dk}_i := \perp$ with overwhelming probability.

Responder-Robustness. Suppose that P_i invokes normal algorithms `Init` and `DerI` while P_r invokes anamorphic algorithm `aDerR`, then P_i computes $m_{i,2} := f_{1,2}(d_{i,2})$ by using a uniformly chosen $d_{i,2} \leftarrow_{\S} \mathcal{D}_{1,2}$. When P_r invokes the anamorphic algorithm `aDerR` to check whether $m_{i,2} = f_{1,2}(\text{PRF}_1(s_r, m_{i,1}))$ holds, we know that here $f_{1,2}(\text{PRF}_1(s_r, m_{i,1}))$ is independent of $m_{i,2} := f_{1,2}(d_{i,2})$ since $d_{i,2} \leftarrow_{\S} \mathcal{D}_{1,2}$ is chosen independently of $s_r, m_{i,1}$ by P_i . Thus for every possible value of $f_{1,2}(\text{PRF}_1(s_r, m_{i,1}))$, the check $m_{i,2} := f_{1,2}(d_{i,2}) =$

$f_{1,2}(\text{PRF}_1(s_r, m_{i,1}))$ can pass with only a negligible probability by the entropy-preserving property of $f_{1,2}$ and due to the randomness of $d_{i,2} \leftarrow_{\S} \mathcal{D}_{1,2}$, and consequently, P_r will set $\text{dk}_r := \perp$ overwhelmingly.

4.3 Security Proofs

We show the strong security of the AM-AKE proposed in Subsect. 4.2.

Theorem 4 (Strong Security of AM-AKE). *Let AKE be a qualified two-pass AKE scheme satisfying the requirements listed in Table 2, and let PRF be a pseudo-random function. Then the AM-AKE constructed in Subsect. 4.2 achieves both the sIND-WM and sPR-DK security.*

The proof of Theorem 4 consists of two parts: the sIND-WM security follows from Lemma 1 and Lemma 2, and the sPR-DK security follows from Lemma 3.

Lemma 1. *For any adversary \mathcal{A} , it holds that $|\Pr[\mathcal{A}(\text{pk}, \text{sk}) = 1] - \Pr[\mathcal{A}(\text{apk}, \text{ask}) = 1]| = 0$, where $(\text{pk}, \text{sk}) \leftarrow \text{Gen}$ and $(\text{apk}, \text{ask}, \text{aux}) \leftarrow \text{aGen}$.*

Proof of Lemma 1. In AM-AKE, the anamorphic key-pair (apk, ask) is generated by SimGen , and thus has the same distribution as the norm pair (pk, sk) generated by Gen , according to the secret extractability of Gen . \square

Lemma 2. *There exists PPT simulator $\text{Sim} = (\text{Siml}, \text{SimR})$, such that for any PPT adversary \mathcal{A} and $N = \text{poly}(\kappa)$, $|\Pr[\text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{sIND-WM}} = 1] - \frac{1}{2}| \leq \text{negl}(\kappa)$.*

Lemma 3. *There exists PPT simulator $\text{Sim} = (\text{Siml}, \text{SimR})$, such that for any PPT adversary \mathcal{A} and $N = \text{poly}(\kappa)$, $|\Pr[\text{Exp}_{\text{AM-AKE}, \mathcal{A}, \text{Sim}, N}^{\text{sPR-DK}} = 1] - \frac{1}{2}| \leq \text{negl}(\kappa)$.*

Due to space limitations, the proofs of Lemma 2 and Lemma 3 are shown in the full version [24]. Here we only present the description of the simulator $\text{Sim} = (\text{Siml}, \text{SimR})$ used in these proofs, and we refer to Subsect. 1.2 for an overview of the proofs.

- $R_i \leftarrow \text{Siml}(\text{apk}_r, \text{ask}_i, \text{aux}_i = \text{msk}_i, R'_i)$: Here R'_i is an internal randomness used in $\overline{\text{aInit}}$, and thus includes $d_{i,1}$ as well as the randomness used in $\overline{\text{Init}}$, denoted by $d_{i,3}$, i.e., $R'_i = (d_{i,1}, d_{i,3})$. This algorithm aims to explain R'_i as a randomness R_i for $\overline{\text{Init}}$. To this end, it computes $s_i := \text{Extract}(\text{msk}_i, \text{apk}_r)$, $m_{i,1} := f_{1,1}(d_{i,1})$, $d_{i,2} := \text{PRF}_1(s_i, m_{i,1})$, and outputs $R_i := (d_{i,1}, d_{i,2}, d_{i,3})$.
- $R_r \leftarrow \text{SimR}(\text{apk}_i, \text{ask}_r, \text{aux}_r = \text{msk}_r, m, R'_r)$: Here R'_r is an internal randomness used in $\overline{\text{aDerR}}$, and thus includes $d_{r,1}$ as well as the randomness used in $\overline{\text{DerR}}$, denoted by $d_{r,3}$, i.e., $R'_r = (d_{r,1}, d_{r,3})$. This algorithm aims to explain R'_r as a randomness R_r for $\overline{\text{DerR}}$. To this end, it parses $m = (m_{i,1}, m_{i,2}, m_{i,3})$, computes $s_r := \text{Extract}(\text{msk}_r, \text{apk}_i)$, $m_{r,1} := f_{R,1}(d_{r,1})$, $d_{r,2} := \text{PRF}_R(s_r, (m_{i,1}, m_{r,1}))$ and outputs $R_r := (d_{r,1}, d_{r,2}, d_{r,3})$.

5 Instantiations of Robust and Strongly-Secure AM-AKE

To instantiate the AM-AKE generic construction proposed in Sect. 4, we can employ any pseudo-random function PRF, and thus we only need to instantiate the underlying *qualified AKE*, i.e., AKE satisfying the requirements in Table 2.

In this section, we will show that the popular SIG+KEM paradigm [17] and three-KEM paradigm [18] for constructing AKE yield *qualified AKE* schemes, as long as the underlying SIG and/or KEM satisfy certain conditions. Then by plugging them into the generic construction in Sect. 4, we immediately obtain concrete AM-AKE schemes achieving initiator-robustness, responder-robustness and strong security. More precisely, in Subsect. 5.1, we show how to obtain qualified AKE and AM-AKE via the SIG+KEM paradigm, and in Subsect. 5.2, we show how to obtain them via the three-KEM paradigm.

5.1 Instantiation from The SIG+KEM Paradigm

Qualified AKE via The SIG+KEM Paradigm. We first recall the SIG+KEM paradigm of constructing two-pass AKE according to [17]. Let $\text{KEM} = (\text{Gen}_{\text{KEM}}, \text{Encap}, \text{Decap})$ be a KEM scheme, $\text{SIG} = (\text{Gen}_{\text{SIG}}, \text{Sign}, \text{Vrfy})$ a signature scheme and H a suitable hash function. The resulting $\text{AKE}_{\text{KS}} = (\text{Gen}_{\text{KS}}, \text{Init}_{\text{KS}}, \text{DerR}_{\text{KS}}, \text{DerL}_{\text{KS}})$ is described as follows (see also Fig. 5 with dotted boxes for the paradigm).

- $(\text{pk}, \text{sk}) \leftarrow \text{Gen}_{\text{KS}} : \text{Invoke } (\text{pk}, \text{sk}) \leftarrow \text{Gen}_{\text{SIG}} \text{ and return } (\text{pk}, \text{sk}).$
- $(\text{msg}_i, \text{st}) \leftarrow \text{Init}_{\text{KS}}(\text{pk}_r, \text{sk}_i) : \text{Invoke } (\widetilde{\text{pk}}, \widetilde{\text{sk}}) \leftarrow \text{Gen}_{\text{KEM}}, \sigma_i \leftarrow \text{Sign}(\text{sk}_i, \widetilde{\text{pk}}), \text{ and output } \text{msg}_i := (\widetilde{\text{pk}}, \sigma_i) \text{ and the state } \text{st} := (\widetilde{\text{pk}}, \widetilde{\text{sk}}).$
- $(\text{msg}_r, K_r) \leftarrow \text{DerR}_{\text{KS}}(\text{pk}_i, \text{sk}_r, \text{msg}_i = (\widetilde{\text{pk}}, \sigma_i)) : \text{If } \text{Vrfy}(\text{pk}_i, \widetilde{\text{pk}}, \sigma_i) = 0: \text{output } \perp; \text{ if } \text{Vrfy}(\text{pk}_i, \widetilde{\text{pk}}, \sigma_i) = 1: \text{invoke } (K, \psi) \leftarrow \text{Encap}(\widetilde{\text{pk}}), \sigma_r \leftarrow \text{Sign}(\text{sk}_r, (\widetilde{\text{pk}}, \psi)), \text{ and output } \text{msg}_r := (\psi, \sigma_r) \text{ and } K_r := H(K, \text{pk}_i, \text{pk}_r, \text{msg}_i, \text{msg}_r).$
- $K_i \leftarrow \text{DerL}_{\text{KS}}(\text{pk}_r, \text{sk}_i, \text{msg}_r = (\psi, \sigma_r), \text{st} = (\widetilde{\text{pk}}, \widetilde{\text{sk}})) : \text{If } \text{Vrfy}(\text{pk}_r, (\widetilde{\text{pk}}, \psi), \sigma_r) = 0: \text{output } \perp; \text{ if } \text{Vrfy}(\text{pk}_r, (\widetilde{\text{pk}}, \psi), \sigma_r) = 1: \text{invoke } K \leftarrow \text{Decap}(\widetilde{\text{sk}}, \psi) \text{ and output } K_i := H(K, \text{pk}_i, \text{pk}_r, \text{msg}_i, \text{msg}_r).$

Below we will show that the AKE_{KS} is *qualified* for constructing AM-AKE, if the underlying SIG and KEM satisfy the following requirements (see also Table 3).

Requirements for $\text{SIG} = (\text{Gen}_{\text{SIG}}, \text{Sign}, \text{Vrfy})$:

- Gen_{SIG} has secret extractability, supported by $(\text{SimGen}, \text{Extract})$ as per Definition 10;
- Sign is 2-separable for generating σ , supported by $(f_5, \overline{\text{Sign}})$ as per Definition 9, i.e., $\text{Sign}(\text{sk}, m)$ generates σ by sampling $d_5 \leftarrow_{\S} \mathcal{D}_5$, computing $\sigma_1 := f_5(d_5)$, invoking $\sigma_2 \leftarrow \overline{\text{Sign}}(\text{sk}, m, d_5)$, and setting $\sigma := (\sigma_1, \sigma_2)$;
- The function f_5 is entropy-preserving as per Definition 8.

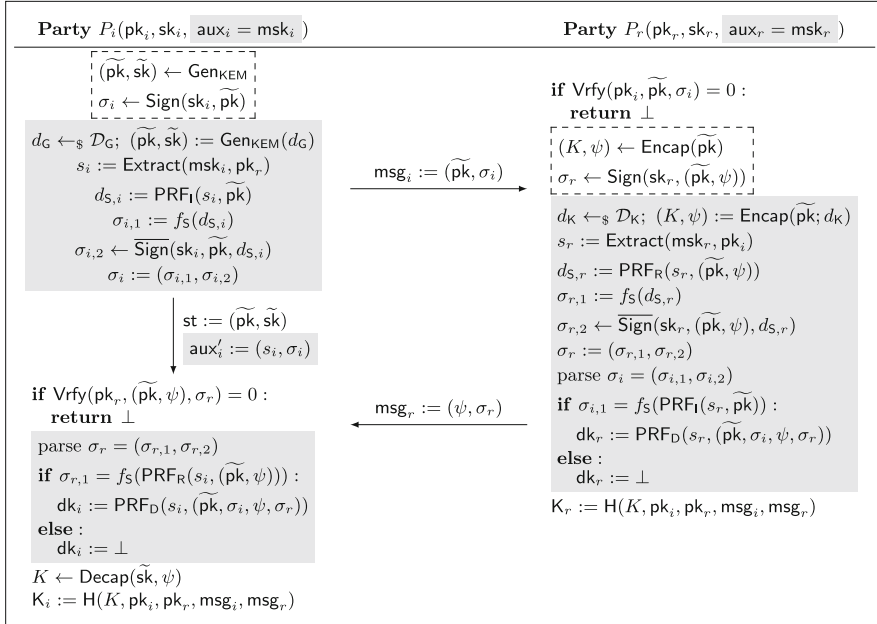


Fig. 5. The SIG+KEM paradigm for AKE (with [dotted boxes]) and the resulting robust and strongly-secure AM-AKE via our generic construction in Sect. 4 (with normal algorithms in [dotted boxes] and anamorphic ones in gray boxes).

Table 3. Requirements for the building blocks $\text{SIG} = (\text{Gen}_{\text{SIG}}, \text{Sign}, \text{Vrfy})$ and $\text{KEM} = (\text{Gen}_{\text{KEM}}, \text{Encap}, \text{Decap})$ of the KEM-SIG paradigm in order to get a qualified AKE_{KS} .

Qualified AKE_{KS}	SIG		KEM	
	Gen_{SIG}	Sign	Gen_{KEM}	Encap
Requirements	<i>secret extract.</i>	<i>2-separable with entropy-preserv. f_S</i>	<i>entropy-preserv.</i>	<i>entropy-preserv.</i>
Supportive Func./Alg.	(SimGen, Extract)	(f_S, Sign)	$\text{pk} := \overline{\text{Gen}}_{\text{KEM}}(d_G)$	$\psi := \overline{\text{Encap}}(d_K)$

Requirements for $\text{KEM} = (\text{Gen}_{\text{KEM}}, \text{Encap}, \text{Decap})$:

- The function $\overline{\text{Gen}}_{\text{KEM}}(\cdot) : \mathcal{D}_G \rightarrow \{0, 1\}^*$ is entropy-preserving, where $\overline{\text{Gen}}_{\text{KEM}}$ functions the same as Gen_{KEM} that takes a randomness $d_G \in \mathcal{D}_G$ as input but outputs only pk (and does not output sk).
- For any public key pk , the function $\overline{\text{Encap}}(\text{pk}; \cdot) : \mathcal{D}_K \rightarrow \{0, 1\}^*$ is entropy-preserving, where $\overline{\text{Encap}}(\text{pk}; \cdot)$ functions the same as $\text{Encap}(\text{pk})$ that takes a randomness $d_K \in \mathcal{D}_K$ as input but outputs only ψ (and does not output K).

With such SIG and KEM, we prove that the resulting AKE_{KS} is a qualified AKE via the following Lemma 4. Then by plugging the qualified AKE_{KS} into our

generic construction in Sect. 4, we immediately get a robust and strongly-secure two-pass AM-AKE scheme, as shown in Fig. 5 with gray boxes.

Lemma 4. *If SIG and KEM meet the above requirements, then the AKE_{KS} yielded by the SIG+KEM paradigm is a qualified AKE for constructing AM-AKE.*

Proof. To prove that $\text{AKE}_{\text{KS}} = (\text{Gen}_{\text{KS}}, \text{Init}_{\text{KS}}, \text{DerR}_{\text{KS}}, \text{DerI}_{\text{KS}})$ is a qualified one, we show that all requirements listed in Table 2 are satisfied, i.e., Gen_{KS} has secret extractability, Init_{KS} is 3-separable with entropy-preserving functions $(f_{1,1}, f_{1,2})$, and DerR_{KS} is 3-separable with entropy-preserving functions $(f_{R,1}, f_{R,2})$.

- Since $\text{Gen}_{\text{KS}} = \text{Gen}_{\text{SIG}}$, the secret extract of Gen_{KS} follows from that of Gen_{SIG} .
- The process of $\text{Init}_{\text{KS}}(\text{pk}_r, \text{sk}_i)$ for generating $(\text{msg}_i = (\widetilde{\text{pk}}, \sigma_i = (\sigma_{i,1}, \sigma_{i,2})), \text{st} = (\widetilde{\text{pk}}, \widetilde{\text{sk}}))$ can be decomposed into three steps, since Sign is 2-separable:
 1. $d_G \leftarrow_{\$} \mathcal{D}_G$ and $\widetilde{\text{pk}} := \overline{\text{Gen}}_{\text{KEM}}(d_G)$. So we can define $f_{1,1} := \overline{\text{Gen}}_{\text{KEM}}$, and then the entropy-preserving property of $f_{1,1}$ follows from that of $\overline{\text{Gen}}_{\text{KEM}}$.
 2. $d_{S,i} \leftarrow_{\$} \mathcal{D}_S$ and $\sigma_{i,1} := f_S(d_{S,i})$. So we can define $f_{1,2} := f_S$, and then the entropy-preserving property of $f_{1,2}$ follows from that of f_S .
 3. $(\widetilde{\text{pk}}, \widetilde{\text{sk}}) := \overline{\text{Gen}}_{\text{KEM}}(d_G)$, $\sigma_{i,2} \leftarrow \overline{\text{Sign}}(\text{sk}_i, \widetilde{\text{pk}}, d_{S,i})$, and set $\text{st} := (\widetilde{\text{pk}}, \widetilde{\text{sk}})$. This process can be defined as $(\sigma_{i,2}, \text{st}) \leftarrow \overline{\text{Init}}_{\text{KS}}(\text{pk}_r, \text{sk}_i, d_G, d_{S,i})$.

Consequently, Init_{KS} is 3-separable with two entropy-preserving functions $(f_{1,1} = \overline{\text{Gen}}_{\text{KEM}}, f_{1,2} = f_S)$ and an algorithm $\overline{\text{Init}}_{\text{KS}}$.

- Similarly, the process of $\text{DerR}_{\text{KS}}(\text{pk}_i, \text{sk}_r, \text{msg}_i = (\widetilde{\text{pk}}, \sigma_i))$ for generating $(\text{msg}_r = (\psi, \sigma_r = (\sigma_{r,1}, \sigma_{r,2})), K_r)$ can be decomposed into three steps:
 1. $d_K \leftarrow_{\$} \mathcal{D}_K$ and $\psi := \overline{\text{Encap}}(\widetilde{\text{pk}}; d_K)$. So we can define $f_{R,1} := \overline{\text{Encap}}(\widetilde{\text{pk}}; \cdot)$, and then the entropy-preserving of $f_{R,1}$ follows from that of $\overline{\text{Encap}}(\widetilde{\text{pk}}; \cdot)$.
 2. $d_{S,r} \leftarrow_{\$} \mathcal{D}_S$ and $\sigma_{r,1} := f_S(d_{S,r})$. So we can define $f_{R,2} := f_S$, and then the entropy-preserving property of $f_{R,2}$ follows from that of f_S .
 3. If $\text{Vrfy}(\text{pk}_i, \widetilde{\text{pk}}, \sigma_i) = 1$: $(K, \psi) := \overline{\text{Encap}}(\widetilde{\text{pk}}; d_K)$, $\sigma_{r,2} \leftarrow \overline{\text{Sign}}(\text{sk}_r, (\widetilde{\text{pk}}, \psi), d_{S,r})$, and set $K_r := H(K, \text{pk}_i, \text{pk}_r, \text{msg}_i, \text{msg}_r)$. Otherwise output \perp . This process can be defined as $(\sigma_{r,2}, K_r) \leftarrow \overline{\text{DerR}}_{\text{KS}}(\text{pk}_i, \text{sk}_r, \text{msg}_i = (\widetilde{\text{pk}}, \sigma_i), d_K, d_{S,r})$.

Consequently, DerR_{KS} is 3-separable with two entropy-preserving functions $(f_{R,1} = \overline{\text{Encap}}(\widetilde{\text{pk}}; \cdot), f_{R,2} = f_S)$ and an algorithm $\overline{\text{DerR}}_{\text{KS}}$. \square

Concrete Instantiations. To obtain concrete qualified AKE scheme via the SIG+KEM paradigm, it remains to present concrete SIG and KEM schemes satisfying the requirements described above (cf. Table 3). More precisely, we will show that any IND-CPA secure KEM suffices, and then for SIG, we present a concrete instantiation over asymmetric pairing groups.

Concrete KEM. In fact, any IND-CPA secure KEM has entropy-preserving $\overline{\text{Gen}}_{\text{KEM}}$ and $\overline{\text{Encap}}$, which output only pk and ψ respectively. Intuitively, if an

independently generated $(\widetilde{\text{pk}}, \widetilde{\text{sk}}) \leftarrow \text{Gen}$ or $(\widetilde{K}, \widetilde{\psi}) \leftarrow \text{Encap}(\text{pk})$ leads to $\widetilde{\text{pk}} = \text{pk}$ or $\widetilde{\psi} = \psi$ with non-negligible probability for a target pk or ψ , an adversary can use the accompanying sk or K to break the IND-CPA security of KEM easily. More precisely, we have the following lemma with proof shown in the full version [24].

Lemma 5 (Any IND-CPA Secure KEM has Entropy-Preserving $\overline{\text{Gen}}_{\text{KEM}}$ and $\overline{\text{Encap}}$). *If $\text{KEM} = (\text{Gen}_{\text{KEM}}, \text{Encap}, \text{Decap})$ is a IND-CPA secure KEM scheme, then the function $\overline{\text{Gen}}_{\text{KEM}}(\cdot)$ that outputs only pk and the function $\overline{\text{Encap}}(\text{pk}; \cdot)$ that outputs only ψ are entropy-preserving.*

Concrete SIG. Let $\text{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2, g_T)$ be a description of asymmetric pairing group, where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are cyclic groups of prime order p , $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a non-degenerated bilinear pairing, and g_1, g_2, g_T are generators of $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ respectively. Moreover, let $\text{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ be a hash function. We present a concrete scheme $\text{SIG}_{\text{DDH}} = (\text{Gen}_{\text{DDH}}, \text{Sign}, \text{Vrfy})$ as follows.

- $(\text{pk}, \text{sk}) \leftarrow \text{Gen}_{\text{DDH}}$: it picks $x \leftarrow_{\S} \mathbb{Z}_p$, and sets $(\text{pk} := e(g_1, g_2)^x, \text{sk} := g_2^x)$.
- $\sigma \leftarrow \text{Sign}(\text{sk} = g_2^x, m)$: it chooses $r \leftarrow_{\S} \mathbb{Z}_p$ randomly, then computes $\sigma_1 := g_1^r$, $d := \text{H}(m, \sigma_1) \in \mathbb{Z}_p$, $\sigma_2 := g_2^{x \cdot d + r}$, and outputs $\sigma := (\sigma_1, \sigma_2)$.
- $0/1 \leftarrow \text{Vrfy}(\text{pk} = e(g_1, g_2)^x, m, \sigma = (\sigma_1, \sigma_2))$: it computes $d := \text{H}(m, \sigma_1) \in \mathbb{Z}_p$, and outputs 1 if and only if $e(g_1, \sigma_2) = e(g_1, g_2)^{x \cdot d} \cdot e(\sigma_1, g_2)$ holds.

Intuitively, the scheme SIG_{DDH} can be viewed as a variant of the Schnorr signature scheme [22], by lifting it from $(\mathbb{Z}_p, \mathbb{G})$ of a cyclic group to $(\mathbb{G}_2, \mathbb{G}_T)$ of the asymmetric pairing group. It is routine to check the correctness of SIG_{DDH} . Next we show its EUF-CMA security with proof shown in the full version [24], since the proof is essentially the same as that for the Schnorr scheme.

Theorem 5 (Security of SIG_{DDH}). *If the DDH assumption holds over \mathbb{G}_1 and H is a random oracle, then the proposed SIG_{DDH} achieves EUF-CMA security.*

Below we show that SIG_{DDH} satisfies the requirements listed in Table 3 via the following two lemmas.

Lemma 6 (Secret Extractability of Gen_{DDH}). *The key generation algorithm Gen_{DDH} has secret extractability based on the DDH assumption over \mathbb{G}_2 .*

Proof. We first describe the supportive algorithms $\text{SimGen}_{\text{DDH}}$ and $\text{Extract}_{\text{DDH}}$ as follows, which take pp as an implicit input, the same as Gen_{DDH} .

- $(\text{pk}, \text{sk}, \text{msk}) \leftarrow \text{SimGen}_{\text{DDH}}$: it picks $x \leftarrow_{\S} \mathbb{Z}_p$, and sets $(\text{pk} := e(g_1, g_2)^x, \text{sk} := g_2^x, \text{msk} := x)$.
- $s \leftarrow \text{Extract}_{\text{DDH}}(\text{msk}_i = x_i, \text{pk}_r = e(g_1, g_2)^{x_r})$: it computes $s := \text{pk}_r^{\text{msk}_i} = e(g_1, g_2)^{x_r x_i}$.

Next we show that the proposed $(\text{SimGen}_{\text{DDH}}, \text{Extract}_{\text{DDH}})$ satisfy the requirements of secret extractability (cf. Definition 10). It is easy to see that the key-pair (pk, sk) generated by $\text{SimGen}_{\text{DDH}}$ has the same distribution as the normal pair generated by Gen_{DDH} , and check that the extraction correctness holds, i.e., $\text{Extract}_{\text{DDH}}(\text{msk}_i, \text{pk}_r) = e(g_1, g_2)^{x_i x_r} = \text{Extract}_{\text{DDH}}(\text{msk}_r, \text{pk}_i)$.

It remains to prove the pseudo-randomness of $\text{Extract}_{\text{DDH}}(\text{msk}_i, \text{pk}_r) = e(g_1, g_2)^{x_i x_r}$ conditioned on $(\text{pk}_i = e(g_1, g_2)^{x_i}, \text{pk}_r = e(g_1, g_2)^{x_r}, \text{sk}_i = g_2^{x_i}, \text{sk}_r = g_2^{x_r})$. More precisely, for any adversary \mathcal{A} against the pseudo-randomness of the extracting, we construct an algorithm \mathcal{B} against the DDH assumption over \mathbb{G}_2 as follows.

Given a DDH challenge $(\text{pp}, g_2^{x_i}, g_2^{x_r}, T)$, \mathcal{B} wants to distinguish $T = g_2^{x_i x_r}$ from $T \leftarrow_{\S} \mathbb{G}_2$, where $x_i, x_r \leftarrow_{\S} \mathbb{Z}_p$. To this end, \mathcal{B} sets $\text{sk}_i := g_2^{x_i}, \text{sk}_r := g_2^{x_r}$, computes $\text{pk}_i := e(g_1, g_2^{x_i}) = e(g_1, g_2)^{x_i}, \text{pk}_r := e(g_1, g_2^{x_r}) = e(g_1, g_2)^{x_r}, s^* := e(g_1, T)$, gives $(\text{pk}_i, \text{pk}_r, \text{sk}_i, \text{sk}_r, s^*)$ to \mathcal{A} , and returns the output of \mathcal{A} to its own challenger. It is easy to see that \mathcal{B} 's simulation of $(\text{pk}_i, \text{pk}_r, \text{sk}_i, \text{sk}_r)$ is perfect. If $T = g_2^{x_i x_r}$, then $s^* := e(g_1, T) = e(g_1, g_2)^{x_i x_r} = \text{Extract}_{\text{DDH}}(\text{msk}_i, \text{pk}_r)$; if $T \leftarrow_{\S} \mathbb{G}_2$, then $s^* := e(g_1, T)$ is uniformly distributed over \mathbb{G}_T . Consequently, \mathcal{B} is able to distinguish $T = g_2^{x_i x_r}$ from $T \leftarrow_{\S} \mathbb{G}_2$, as long as \mathcal{A} can distinguish $(\text{pk}_i, \text{pk}_r, \text{sk}_i, \text{sk}_r, s^* = \text{Extract}_{\text{DDH}}(\text{msk}_i, \text{pk}_r))$ from $(\text{pk}_i, \text{pk}_r, \text{sk}_i, \text{sk}_r, s^* \leftarrow_{\S} \mathbb{G}_T)$, and we have $\text{Adv}_{\text{Gen}_{\text{DDH}}, \mathcal{A}}^{\text{PR-Ext}}(\kappa) \leq \text{Adv}_{\mathbb{G}_2, \mathcal{B}}^{\text{DDH}}(\kappa)$, which is negligible under the DDH assumption over \mathbb{G}_2 . This shows the pseudo-randomness of the extracting. \square

Lemma 7 (2-Separability of Sign_{DDH} with Entropy-Preserving f_S). *Sign_{DDH} is 2-separable for generating σ , and the supportive function f_S is entropy-preserving.*

Proof. It is easy to see that the process of $\text{Sign}(\text{sk} = g_2^x, m)$ generating $\sigma = (\sigma_1, \sigma_2)$ can be decomposed into two parts: the first part includes $r \leftarrow_{\S} \mathbb{Z}_p$ and $\sigma_1 := g_1^r$, and the second part computes $\sigma_2 := g_2^{x \cdot \text{H}(m, \sigma_1) + r}$. Consequently, Sign is 2-separable for generating $\sigma = (\sigma_1, \sigma_2)$, supported by $(f_S, \widetilde{\text{Sign}})$, where f_S is defined by $f_S(r) := g_1^r$ for $r \in \mathbb{Z}_p$ and $\widetilde{\text{Sign}}$ is defined by $\widetilde{\text{Sign}}(\text{sk} = g_2^x, m, r) := g_2^{x \cdot \text{H}(m, g_1^r) + r}$. Moreover, f_S is entropy-preserving since for any $h \in \mathbb{G}_1$, the probability $\Pr[f_S(r) = g_1^r = h | r \leftarrow_{\S} \mathbb{Z}_p] = 1/p$ is negligible. \square

5.2 Instantiation from The Three-KEM Paradigm

Qualified AKE via The Three-KEM Paradigm. We first recall the three-KEM paradigm of constructing two-pass AKE according to [18]. Let $\text{KEM} = (\text{Gen}_{\text{KEM}}, \text{Encap}, \text{Decap})$ and $\text{KEM}_0 = (\text{Gen}_{\text{KEM}_0}, \text{Encap}_0, \text{Decap}_0)$ be two KEM schemes, and H a suitable hash function. The resulting $\text{AKE}_{3\text{K}} = (\text{Gen}_{3\text{K}}, \text{Init}_{3\text{K}}, \text{DerR}_{3\text{K}}, \text{DerL}_{3\text{K}})$ is described as follows (see also Fig. 6 with dotted boxes).

- $(\text{pk}, \text{sk}) \leftarrow \text{Gen}_{3\text{K}}$: Invoke $(\text{pk}, \text{sk}) \leftarrow \text{Gen}_{\text{KEM}}$ and return (pk, sk) .
- $(\text{msg}_i, \text{st}) \leftarrow \text{Init}_{3\text{K}}(\text{pk}_r, \text{sk}_i)$: Invoke $(\widetilde{\text{pk}}, \widetilde{\text{sk}}) \leftarrow \text{Gen}_{\text{KEM}_0}, (K_i, \psi_i) \leftarrow \text{Encap}(\text{pk}_r)$, and output $\text{msg}_i := (\widetilde{\text{pk}}, \psi_i)$ and the state $\text{st} := (\widetilde{\text{sk}}, K_i)$.

- $(\text{msg}_r, K_r) \leftarrow \text{DerR}_{3K}(\text{pk}_i, \text{sk}_r, \text{msg}_i = (\widetilde{\text{pk}}, \psi_i))$: Invoke $K_i \leftarrow \text{Decap}(\text{sk}_r, \psi_i)$, $(\widetilde{K}, \widetilde{\psi}) \leftarrow \text{Encap}_0(\widetilde{\text{pk}})$ and $(K_r, \psi_r) \leftarrow \text{Encap}(\text{pk}_i)$. Output $\text{msg}_r := (\widetilde{\psi}, \psi_r)$ and session key $K_r := \text{H}(\text{pk}_i, \text{pk}_r, \text{msg}_i, \text{msg}_r, K_i, K_r, \widetilde{K})$.
- $K_i \leftarrow \text{Derl}_{3K}(\text{pk}_r, \text{sk}_i, \text{msg}_r = (\psi, \psi_r), \text{st} = (\text{sk}, K_i))$:
Invoke $\widetilde{K} \leftarrow \text{Decap}_0(\widetilde{\text{sk}}, \widetilde{\psi})$, $K_r \leftarrow \text{Decap}(\text{sk}_i, \psi_r)$, and output $K_i := \text{H}(\text{pk}_i, \text{pk}_r, \text{msg}_i, \text{msg}_r, K_i, K_r, \widetilde{K})$.

Table 4. Requirements for the building blocks $\text{KEM} = (\text{Gen}_{\text{KEM}}, \text{Encap}, \text{Decap})$ and $\text{KEM}_0 = (\text{Gen}_{\text{KEM}_0}, \text{Encap}_0, \text{Decap}_0)$ of the three-KEM paradigm in order to get a qualified AKE_{3K} .

Qualified AKE_{3K}	KEM		KEM_0	
	Gen_{KEM}	Encap	$\text{Gen}_{\text{KEM}_0}$	Encap_0
Requirements	<i>secret extract.</i>	<i>entropy-preserv.</i>	<i>entropy-preserv.</i>	<i>entropy-preserv.</i>
Supportive Func./Alg.	(SimGen, Extract)	$\psi := \overline{\text{Encap}}(d_K)$	$\widetilde{\text{pk}} := \overline{\text{Gen}}_{\text{KEM}}(d_G)$	$\widetilde{\psi} := \overline{\text{Encap}}(d_{K_0})$

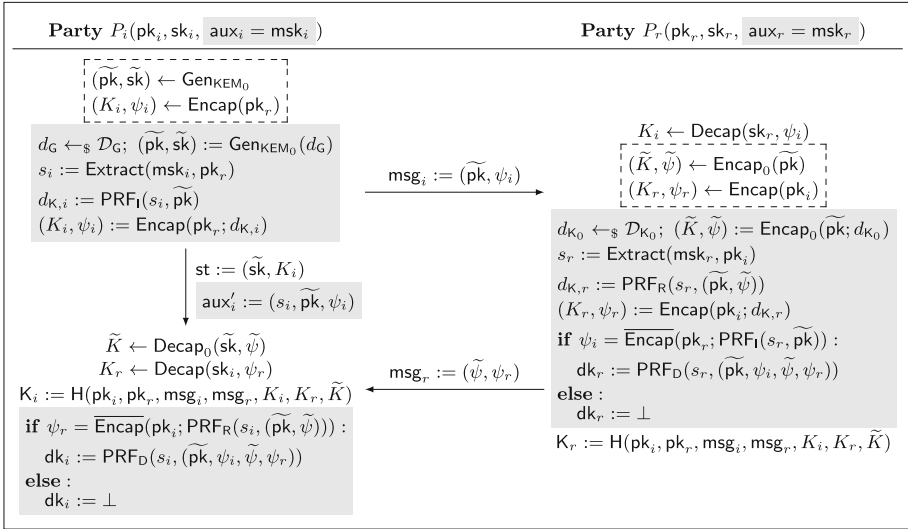


Fig. 6. The three-KEM paradigm for AKE (with [dotted boxes]) and the resulting robust and strongly-secure AM-AKE via our generic construction in Sect. 4 (with normal algorithms in [dotted boxes] and anamorphic ones in gray boxes).

Below we will show that AKE_{3K} is *qualified* for constructing AM-AKE, if the underlying KEM and KEM_0 satisfy the following requirements (see also Table 4).

Requirements for KEM = (Gen_{KEM}, Encap, Decap) :

- Gen_{KEM} has secret extractability, supported by (SimGen, Extract) as per Definition 10;
- For any public key \mathbf{pk} , the function $\overline{\text{Encap}}(\mathbf{pk}; \cdot) : \mathcal{D}_K \longrightarrow \{0, 1\}^*$ is entropy-preserving, where $\overline{\text{Encap}}(\mathbf{pk}; \cdot)$ functions the same as Encap(\mathbf{pk}) that takes a randomness $d_K \in \mathcal{D}_K$ as input but outputs only ψ (and does not output K).

Requirements for KEM₀ = (Gen_{KEM₀}, Encap₀, Decap₀) :

- The function $\overline{\text{Gen}}_{\text{KEM}_0}(\cdot) : \mathcal{D}_G \longrightarrow \{0, 1\}^*$ is entropy-preserving, where $\overline{\text{Gen}}_{\text{KEM}_0}$ functions the same as Gen_{KEM₀} that takes a randomness $d_G \in \mathcal{D}_G$ as input but outputs only $\widetilde{\mathbf{pk}}$ (and does not output $\widetilde{\mathbf{sk}}$).
- For any public key $\widetilde{\mathbf{pk}}$, the function $\overline{\text{Encap}}_0(\widetilde{\mathbf{pk}}; \cdot) : \mathcal{D}_{K_0} \longrightarrow \{0, 1\}^*$ is entropy-preserving, where $\overline{\text{Encap}}_0(\widetilde{\mathbf{pk}}; \cdot)$ is the same as Encap₀($\widetilde{\mathbf{pk}}$) that takes a randomness $d_{K_0} \in \mathcal{D}_{K_0}$ as input but outputs only $\widetilde{\psi}$ (and does not output \widetilde{K}).

With such KEM and KEM₀, we prove that the resulting AKE_{3K} is a qualified AKE via the following Lemma 8. The proof of Lemma 8 is quite similar to that of Lemma 4 in Subsect. 5.1, and thus we show it in the full version [24].

Lemma 8. *If KEM and KEM₀ meet the above requirements, the AKE_{3K} yielded by the three-KEM paradigm is a qualified AKE for constructing AM-AKE.*

Then by plugging the qualified AKE_{3K} into our generic construction in Sect. 4, we immediately get a robust and strongly-secure two-pass AM-AKE scheme, as shown in Fig. 6 with gray boxes.

Concrete Instantiations. To obtain concrete qualified AKE scheme via the three-KEM paradigm, it remains to present concrete KEM schemes KEM and KEM₀ satisfying the requirements described above (cf. Table 4). Specially, as shown in Lemma 5 in Subsect. 5.1, any IND-CPA secure KEM has entropy-preserving $\overline{\text{Gen}}_{\text{KEM}}$ and $\overline{\text{Encap}}$, so we can instantiate KEM₀ with any IND-CPA secure KEM scheme. For KEM, we present a concrete instantiation over asymmetric pairing groups.

Concrete KEM scheme KEM. Let $\mathbf{pp} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2, g_T)$ be a description of asymmetric pairing group. We present a concrete KEM scheme KEM_{DDH} = (Gen_{DDH}, Encap, Decap) as follows:

- $(\mathbf{pk}, \mathbf{sk}) \leftarrow \text{Gen}_{\text{DDH}}$: it picks $x \leftarrow_{\S} \mathbb{Z}_p$, and sets $(\mathbf{pk} := e(g_1, g_2)^x, \mathbf{sk} := g_2^x)$.
- $(K, \psi) \leftarrow \overline{\text{Encap}}(\mathbf{pk} = e(g_1, g_2)^x)$: it chooses $r \leftarrow_{\S} \mathbb{Z}_p$ randomly, then computes $\psi := g_1^r$, $K := (e(g_1, g_2)^x)^r = e(g_1, g_2)^{xr}$, and outputs (K, ψ) .
- $K \leftarrow \overline{\text{Decap}}(\mathbf{sk} = g_2^x, \psi = g_1^r)$: it computes $K := e(g_1^r, g_2^x)$ and outputs K .

It is routine to check the correctness of KEM_{DDH}. Next we show its IND-CPA security based on the DDH assumption over \mathbb{G}_1 via the following theorem. The proof is quite straightforward and thus we show it in the full version [24].

Theorem 6 (Security of KEM_{DDH}). *If the DDH assumption holds over \mathbb{G}_1 , then the proposed KEM_{DDH} achieves IND-CPA security.*

Below we show that KEM_{DDH} satisfies the requirements listed in Table 4, i.e., its key generation algorithm Gen_{DDH} has secret extractability, and the function $\text{Encap}(\text{pk}; \cdot)$ that outputs only ψ is entropy-preserving. Since Gen_{DDH} is identical to that of SIG_{DDH} in Subsect. 5.1, as shown in Lemma 6, Gen_{DDH} has secret extractability under the DDH assumption over \mathbb{G}_2 . Moreover, by Lemma 5, the function $\text{Encap}(\text{pk}; \cdot)$ is entropy-preserving by the IND-CPA security of KEM_{DDH} .

Acknowledgements. We would like to thank the reviewers for their valuable comments. The authors were partially supported by National Natural Science Foundation of China (Grant Nos. 61925207, 62372292), Guangdong Major Project of Basic and Applied Basic Research (2019B030302008), the National Key R&D Program of China under Grant 2022YFB2701500, and Young Elite Scientists Sponsorship Program by China Association for Science and Technology (YESS20200185).

References

1. Banfi, F., Gegier, K., Hirt, M., Maurer, U., Rito, G.: Anamorphic encryption, revisited. In: Joye, M., Leander, G. (eds.) *Advances in Cryptology – EUROCRYPT 2024*. pp. 3–32. Springer Nature Switzerland, Cham (2024)
2. Bellare, M., Rogaway, P.: Optimal asymmetric encryption. In: Santis, A.D. (ed.) *EUROCRYPT’94*. LNCS, vol. 950, pp. 92–111. Springer, Heidelberg (May 1995). <https://doi.org/10.1007/BFb0053428>
3. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J. (eds.) *EUROCRYPT 2004*. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_4
4. Boyd, C., Cliff, Y., González Nieto, J., Paterson, K.G.: Efficient one-round key exchange in the standard model. In: Mu, Y., Susilo, W., Seberry, J. (eds.) *ACISP 08*. LNCS, vol. 5107, pp. 69–83. Springer, Heidelberg (Jul 2008)
5. Catalano, D., Giunta, E., Migliaro, F.: Anamorphic encryption: New constructions and homomorphic realizations. In: Joye, M., Leander, G. (eds.) *Advances in Cryptology – EUROCRYPT 2024*. pp. 33–62. Springer Nature Switzerland, Cham (2024)
6. Diffie, W., Hellman, M.E.: New directions in cryptography. *IEEE Transactions on Information Theory* **22**(6), 644–654 (1976). <https://doi.org/10.1109/TIT.1976.1055638>
7. 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
8. Goldwasser, S., Micali, S.: Probabilistic encryption. *Journal of Computer and System Sciences* **28**(2), 270–299 (1984). [https://doi.org/10.1016/0022-0000\(84\)90070-9](https://doi.org/10.1016/0022-0000(84)90070-9), <https://www.sciencedirect.com/science/article/pii/0022000084900709>
9. 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

10. Harkins, D., Carrel, D.: The Internet Key Exchange (IKE). IETF RFC 2409 (Proposed Standard) (1998)
11. Horel, T., Park, S., Richelson, S., Vaikuntanathan, V.: How to subvert backdoored encryption: Security against adversaries that decrypt all ciphertexts. In: Blum, A. (ed.) ITCS 2019. vol. 124, pp. 42:1–42:20. LIPIcs (Jan 2019). <https://doi.org/10.4230/LIPIcs.ITCS.2019.42>
12. 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
13. 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.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 759–790. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_25
14. Kutylowski, M., Persiano, G., Phan, D.H., Yung, M., Zawada, M.: The self-anticensorship 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>
15. 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
16. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT'99. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (May 1999). https://doi.org/10.1007/3-540-48910-X_16
17. Pan, J., Qian, C., Ringerud, M.: Signed (group) Diffie-Hellman key exchange with tight security. Journal of Cryptology **35**(4), 26 (Oct 2022). <https://doi.org/10.1007/s00145-022-09438-y>
18. 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
19. Pan, J., Wagner, B., Zeng, R.: Tighter security for generic authenticated key exchange in the QROM. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part IV. LNCS, vol. 14441, pp. 401–433. Springer, Heidelberg (Dec 2023). https://doi.org/10.1007/978-981-99-8730-6_13
20. Persiano, G., Phan, D.H., Yung, M.: Anamorphic encryption: Private communication against a dictator. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 34–63. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_2
21. Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446 (Aug 2018). <https://doi.org/10.17487/RFC8446>, <https://www.rfc-editor.org/info/rfc8446>
22. 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
23. von Ahn, L., Hopper, N.J.: Public-key steganography. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 323–341. Springer, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_20

24. Wang, W., Han, S., Liu, S.: Anamorphic authenticated key exchange: Double key distribution under surveillance. *Cryptology ePrint Archive*, Paper 2024/1438 (2024), <https://eprint.iacr.org/2024/1438>
25. Wang, Y., Chen, R., Huang, X., Yung, M.: Sender-anamorphic encryption reformulated: Achieving robust and generic constructions. In: Guo, J., Steinfeld, R. (eds.) *ASIACRYPT 2023, Part VI*. LNCS, vol. 14443, pp. 135–167. Springer, Heidelberg (Dec 2023). https://doi.org/10.1007/978-981-99-8736-8_5
26. Xiao, Y., Zhang, R., Ma, H.: Tightly secure two-pass authenticated key exchange protocol in the CK model. In: Jarecki, S. (ed.) *CT-RSA 2020*. LNCS, vol. 12006, pp. 171–198. Springer, Heidelberg (Feb 2020). https://doi.org/10.1007/978-3-030-40186-3_9

Succinct Arguments



RoK, Paper, SISsors Toolkit for Lattice-Based Succinct Arguments (Extended Abstract)

Michael Kloöß¹, Russell W. F. Lai², Ngoc Khanh Nguyen³,
and Michał Osadnik²(✉)

¹ ETH Zurich, Zurich, Switzerland
michael.klooss@inf.ethz.ch

² Aalto University, Espoo, Finland
michal.osadnik@aalto.fi

³ King's College London, London, UK

Abstract. Lattice-based succinct arguments allow to prove bounded-norm satisfiability of relations, such as $f(\mathbf{s}) = \mathbf{t} \bmod q$ and $\|\mathbf{s}\| \leq \beta$, over specific cyclotomic rings $\mathcal{O}_{\mathcal{K}}$, with proof size polylogarithmic in the witness size. However, state-of-the-art protocols require either 1) a super-polynomial size modulus q due to a soundness gap in the security argument, or 2) a verifier which runs in time linear in the witness size. Furthermore, construction techniques often rely on specific choices of \mathcal{K} which are not mutually compatible. In this work, we exhibit a diverse toolkit for constructing efficient lattice-based succinct arguments:

- (i) We identify new subtractive sets for general cyclotomic fields \mathcal{K} and their maximal real subfields \mathcal{K}^+ , which are useful as challenge sets, e.g. in arguments for exact norm bounds.
- (ii) We construct modular, verifier-succinct reductions of knowledge for the bounded-norm satisfiability of structured-linear/inner-product relations, without any soundness gap, under the vanishing SIS assumption, over any \mathcal{K} which admits polynomial-size subtractive sets.
- (iii) We propose a framework to use twisted trace maps, i.e. maps of the form $\tau(z) = \frac{1}{N} \cdot \text{Trace}_{\mathcal{K}/\mathbb{Q}}(\alpha \cdot z)$, to embed \mathbb{Z} -inner-products as \mathcal{R} -inner-products for some structured subrings $\mathcal{R} \subseteq \mathcal{O}_{\mathcal{K}}$ whenever the conductor has a square-free odd part.
- (iv) We present a simple extension of our reductions of knowledge for proving the consistency between the coefficient embedding and the Chinese Remainder Transform (CRT) encoding of \mathbf{s} over any cyclotomic field \mathcal{K} with a smooth conductor, based on a succinct decomposition of the CRT map into automorphisms, and a new, simple succinct argument for proving automorphism relations.

Combining all techniques, we obtain, for example, verifier-succinct arguments for proving that \mathbf{s} satisfying $f(\mathbf{s}) = \mathbf{t} \bmod q$ has binary coefficients, without soundness gap and with polynomial-size modulus q .

M. Kloöß—Work done at Aalto University. The author's affiliation changed before publication.

1 Introduction

A fundamental and recurring task in constructing lattice-based succinct arguments is to prove knowledge of a committed vector $\mathbf{s} \in \mathcal{R}^m$ over a ring \mathcal{R} which satisfies norm-bound constraints, such as $\|\mathbf{s}\| \leq \beta$. For instance, such protocols could be extended directly into a succinct argument for structured languages [11], combined with quadratic functional commitments to yield succinct arguments for NP [1, 11]¹, or transformed into polynomial commitment schemes [2, 12, 16] which allow compiling polynomial interactive oracle proofs into succinct arguments.

As evidenced in prior works [7, 22, 24], the currently most efficient lattice-based (non-)succinct arguments operate over rings of integers $\mathcal{R} := \mathbb{Z}[\zeta]$ of cyclotomic number fields $\mathcal{K} := \mathbb{Q}(\zeta)$, where ζ is a primitive f -th root of unity for $f = \text{poly}$. Indeed, the ability to construct exponential-sized low-norm challenge sets over \mathcal{R} allows the aforementioned protocols to achieve negligible soundness in one-shot while maintaining relatively small lattice parameters. However, this comes at a cost of the following two complications.

Correctness Gap. The first one can be described as the *correctness gap*. Namely, most of the recursion-based protocols start with the initial witness $\mathbf{s}_0 := \mathbf{s}$, and in the i -th iteration, an honest prover somehow folds the “current” witness \mathbf{s}_{i-1} into a new one \mathbf{s}_i ; thus shrinking the dimension of the witness, but simultaneously, increasing its norm. At the end, say after μ iterations, the prover outputs the final witness \mathbf{s}_μ of small (potentially constant) dimension. Suppose there exists some γ such that for all $i = 1, \dots, \mu$ we have $\|\mathbf{s}_i\| \leq \gamma \cdot \|\mathbf{s}_{i-1}\|$. Then, in order to maintain correctness, one must inherently choose $q > \gamma^\mu \cdot \beta \geq \|\mathbf{s}_\mu\|$. We call this phenomenon the correctness gap, since if our only task were to commit to \mathbf{s} using a standard lattice-based commitment scheme, setting $q = O(\beta)$ would suffice².

Soundness Gap. A more concerning issue is the *soundness gap*. A vast majority of prior works based on cyclotomic rings encounter the problem that the extracted witness $\bar{\mathbf{s}}$ is not necessarily short, but it is of the fractional form $\bar{\mathbf{s}} := \bar{\mathbf{z}}/\bar{c} \bmod q$, where q is the proof system modulus and both $\bar{\mathbf{z}} \in \mathcal{R}^m$ and $\bar{c} \in \mathcal{R}$ are somewhat short (but $\|\bar{\mathbf{z}}\|$ is larger than β). Even though this *relaxed* soundness suffices to construct basic primitives, such as signature schemes [14, 22], verifiable encryption [23], or few-time verifiable random functions [15], it is not enough when the required functionality naturally involves proving exact norm bounds (e.g. in set membership and range proofs). But especially in the context of succinct arguments built in a recursive manner, dealing with the slack and other norm-growth related issues have shown to have enormous impact on setting up the parameters [2, 3, 10], such as picking super-polynomial modulus q , which makes the aforementioned schemes seem barely practical.

¹ [1, 11] relied on the knowledge-kRISIS assumption for the knowledge soundness of well-formedness of commitments. However, the assumption has subsequently been cryptanalysed [13, 29], rendering the security proofs vacuous.

² For presentation, we omitted the factors related to the security parameter λ .

Prior Works. Since the soundness gap seemed to be the main efficiency bottleneck of lattice-based succinct arguments, several works naturally tried to address this issue first. To begin with, Albrecht and Lai [3] designed a lattice-based argument of polylogarithmic size, where the extracted witness $\bar{\mathbf{s}}$ is somewhat short. The key ingredient of [3] was the notion of *subtractive sets*. Namely, a set $S \subseteq \mathcal{R}$ is called subtractive if for any two distinct elements $c, c' \in S$, $c - c'$ is invertible over the ring \mathcal{R} . Since the invertibility is independent of the proof system modulus q , the latter can be picked freely so that the inverse $(c - c')^{-1}$ is short relative to q . Further, it was shown how to construct such subtractive sets of cardinality p in cyclotomic rings of prime power conductors $\mathfrak{f} := p^k$. Thus, using subtractive sets as a challenge space for the verifier, one can argue that the extracted witness $\bar{\mathbf{s}} := \bar{\mathbf{z}}/\bar{c}$ has low norm, because $1/\bar{c}$ itself is short. However, this approach comes at a cost of non-negligible soundness error (due to the size of subtractive sets), and therefore some sort of soundness amplification is necessary. Furthermore, the protocol itself still does not manage to prove the exact norm bound, i.e. $\|\mathbf{s}\| \leq \beta$. In fact, in the context of recursive succinct arguments, the norm of the extracted witness can only be upper bounded by $\gamma^\mu \cdot \theta^{O(\mu)} \cdot \beta$ for some $\theta \approx \mathfrak{f}$.

In the setting of power-of-two cyclotomic rings, the strategy above falls apart completely since there exists no subtractive set of size larger than two [3, 21]. Hence, a different methodology has recently been developed. Notably, Beullens and Seiler [7] proposed a succinct argument, LaBRADOR, for proving $\|\mathbf{s}\|^2 \leq \beta^2$ (among other relations), inspired by the following two-fold approach from [24]:

- (i) *Approximate shortness proof.* Prove that \mathbf{s} is somewhat short.
- (ii) *\mathbb{Z}_q -Inner product proof.* Prove that $\langle \psi(\mathbf{s}), \psi(\mathbf{s}) \rangle \pmod{q} \leq \beta^2$, where $\psi(\mathbf{s})$ is the coefficient vector of \mathbf{s} .

Combining (i) and (ii), one can argue that for a large enough modulus q no modulo wrap-around occurs, and therefore $\langle \psi(\mathbf{s}), \psi(\mathbf{s}) \rangle \leq \beta^2$ holds over \mathbb{Z} .

In order to prove (i) without relying on subtractive sets, LaBRADOR uses the Johnson-Lindenstrauss random projection technique [4, 17, 26]. The idea is that the verifier will first generate an integer matrix \mathbf{B} with short (binary or ternary) values as a challenge, and the prover then outputs $\psi(\mathbf{v}) := \mathbf{B}\psi(\mathbf{s}) \pmod{q}$. Afterwards, the verifier checks whether $\psi(\mathbf{v})$ is of low norm (which is true in the honest executions, since both \mathbf{B} and $\psi(\mathbf{s})$ are). Finally, the prover needs to prove wellformedness of $\psi(\mathbf{v})$, i.e. the linear equation $\mathbf{B}\psi(\mathbf{s}) = \psi(\mathbf{v})$ over \mathbb{Z}_q . The crucial soundness argument is that if the extracted \mathbf{s} was not short, then with high probability (dictated by the number of rows of \mathbf{B}), $\psi(\mathbf{v}) = \mathbf{B}\psi(\mathbf{s})$ would not have low norm, which leads to a contradiction. Unfortunately, the random projection strategy inherently requires the verifier to generate the matrix \mathbf{B} , which itself has length $O(m)$. As a consequence, the verifier runtime becomes essentially linear in the witness size, which may not be satisfying in certain real-world use cases.

We highlight that both (i) and (ii) require some kind of inner product proof over \mathbb{Z}_q ; either between two committed vectors, or between one public and one committed vector. Since the underlying protocol natively operates over cyclotomic rings $\mathcal{R} = \mathbb{Z}[\zeta]$, it is essential to transform \mathbb{Z} -relations into equivalent ones

over the ring \mathcal{R} . To this end, it was shown in [24] that for any two elements $a, b \in \mathcal{R}$ of a power-of-two cyclotomic ring, the constant term³ of $a \cdot \bar{b} \in \mathcal{R}$ is exactly equal to the inner product $\langle \psi(\mathbf{a}), \psi(\mathbf{b}) \rangle \in \mathbb{Z}$, where $\psi(\mathbf{a}), \psi(\mathbf{b})$ are the coefficient vectors of a, b respectively and $\bar{\cdot}$ here denotes the complex conjugation. This observation allows us to translate proving inner products and linear relations over integers into proving statements about constant terms over the ring \mathcal{R} . Finally, LaBRADOR makes use of the fact that inner product relations over \mathcal{R} are “folding-friendly” and can be efficiently proven in a recursive manner.

Interestingly, LaBRADOR also managed to circumvent the correctness gap by taking inspiration from the “decompose-then-hash” paradigm used in lattice-based Merkle trees [28]. Intuitively, using the notation above for describing recursive-based protocols, instead of folding the intermediate witness \mathbf{s}_{i-1} directly into a new one \mathbf{s}_i , an honest prover would first decompose \mathbf{s}_{i-1} (w.r.t. some decomposition base b) into multiple vectors $(\mathbf{s}_{i-1,j})_{j \in [\ell]}$ of much smaller norm and then fold all of them together into a new witness \mathbf{s}_i ⁴. By carefully picking various parameters, such as b , one can ensure that, in an honest execution, if $\|\mathbf{s}_{i-1}\| \leq \beta$, then we must have $\|\mathbf{s}_i\| \leq \beta$. This technique was also adopted in a recent folding scheme called LatticeFold [8].

Bridging the Gap. At a high level, the aforementioned approaches to prove shortness seem somewhat orthogonal. For $\mathfrak{f} = p^k$, where $p = \text{poly}$ is a large enough prime, one can rely on subtractive sets to efficiently prove approximate shortness **1** with succinct verification [11]. However, it is unknown how to translate proving \mathbb{Z}_q -relations, as in **1**, into equivalent relations over odd prime-power cyclotomic rings. On the other hand, for $\mathfrak{f} = 2^k$, one can apply the Johnson-Lindenstrauss projection strategy to prove both **1** and **1**, but at the cost of slow verification time.

Hence, it is an important research question whether there exist cyclotomic (or other) rings \mathcal{R} , which contain subtractive sets of fairly large size, and at the same time, expose efficient packing and batching techniques for turning relations over \mathbb{Z} (or more generally, other base rings) to relations over \mathcal{R} . An affirmative answer, together with existing optimisations, would then yield a practical lattice-based succinct argument for proving exact norm bounds with fast verification.

1.1 Our Contributions

In this work, we present a versatile toolkit for constructing lattice-based succinct arguments that eliminate correctness and soundness gaps while maintaining succinct verification. Our contributions are outlined as follows:

Succinct Arguments for Bounded-Norm Satisfiability. We design a lattice-based succinct argument system for bounded-norm satisfiability of structured linear and inner-product relations. Our system retains features of previous protocols, such as transparent setup, quasi-linear-time prover, and

³ We say that $a_0 \in \mathbb{Z}$ is the constant term of the ring element $a = \sum_{i=0}^{\varphi(\mathfrak{f})} a_i \zeta^i \in \mathbb{Z}[\zeta]$.

⁴ For soundness, the prover needs to prove additional relations involving $(\mathbf{s}_{i-1,j})_{j \in [\ell]}$.

polylogarithmic-time verifier, while simultaneously eliminating any correctness and soundness gaps. Consequently, our argument system achieves asymptotically the most attractive proof sizes, which are smaller by at least a factor of $\Omega(\log^2 \lambda)$ smaller than the prior state-of-the-art constructions (see Fig. 1 for more details). Furthermore, our protocol’s modular design allows for straightforward analysis and customisation, making it adaptable to various applications.

Subtractive Sets. Our protocol uses subtractive sets as challenge sets. While subtractive sets for prime-power cyclotomic rings are well-known, the non-prime-power case seems less studied. Motivated by the need of non-prime-power rings (e.g. for the twisted trace technique, see below) in some applications, we identify a subtractive set for cyclotomic rings $\mathbb{Z}[\zeta_f]$ of non-prime-power conductor f with a cardinality of f/f_{\max} , where f_{\max} is the largest prime-power divisor of f . Additionally, we identify subtractive sets over the real subrings $\mathbb{Z}[\zeta_f + \zeta_f^{-1}]$, with a cardinality of $(p+1)/2$ for prime-power conductors $f = p^k$ and $\lfloor f/(2f_{\max}) \rfloor$ for non-prime-power f .

Embedded \mathbb{Z} -Inner-Products via Twisted Trace. While our protocol supports proving inner products over rings such as $\mathbb{Z}[\zeta_f]$, higher-layer applications may require proving inner products over \mathbb{Z} , e.g. for proving that a committed \mathbb{Z} -vector is binary. Unfortunately, efficient methods for embedding \mathbb{Z} -inner products to $\mathbb{Z}[\zeta_f]$ -inner products were only known for $f = 2^d$ being a power of 2, which is problematic because subtractive sets over $\mathbb{Z}[\zeta_{2^d}]$ are of cardinality at most 2. We extend the existing embedding method to any ring of the form $\mathbb{Z}[\zeta_{2^d}] \otimes \mathbb{Z}[\zeta_{p_0} + \zeta_{p_0}^{-1}] \otimes \dots \otimes \mathbb{Z}[\zeta_{p_{k-1}} + \zeta_{p_{k-1}}^{-1}]$, where p_0, \dots, p_{k-1} are distinct odd primes. This is achieved by replacing the “constant term map” with a “twisted trace map” defined as: $\tau(z) = \frac{1}{N} \text{Trace}(\alpha \cdot z)$.

Succinct Consistency Proof for CRT. Another typical way of embedding \mathbb{Z} -relations into $\mathbb{Z}[\zeta_f]$ -relations is via the Chinese Remainder Transform (CRT). However, this requires proving that the witness vector is committed in both the coefficient embedding and its CRT coefficients consistently, and known consistency proofs are not succinct. Using the fact that the CRT over cyclotomic fields with smooth conductors can be succinctly represented through a few automorphism evaluations, we derive a succinct argument for the consistency between the commitment of the coefficient embedding and that of the CRT coefficients. At the core of our succinct consistency proof is a new succinct argument that verifies whether two committed vectors are related by an entry-wise automorphism. Due to space constraints, we refer to the full version of this work for detailed CRT-related results.

2 Technical Overview

Throughout this work, we will assume that $\mathcal{K} = \mathbb{Q}(\zeta)$ is a cyclotomic field with conductor f and degree $\varphi = \varphi(f) = \text{poly}$, and $\mathcal{O}_{\mathcal{K}} = \mathbb{Z}[\zeta]$ is its ring of integers. For some of our results, we will further require $\mathcal{K}^+ = \mathbb{Q}(\zeta + \zeta^{-1})$, the maximal real subfield of \mathcal{K} , and its ring of integers $\mathcal{O}_{\mathcal{K}^+} = \mathbb{Z}[\zeta + \zeta^{-1}]$. Depending on the

scheme	assumptions	proof size
[9]	M-SIS	$O(\log^6 m \cdot \lambda^2 / \log \lambda)$
[11]	vSIS	$O(\log^5 m \cdot \lambda^2 / \log^2 \lambda)$
[16]	PowerBASIS	$O(\log^5 m \cdot \lambda^2 / \log^2 \lambda)$
[2]	M-SIS	$O(\log^5 m \cdot \lambda^2 / \log^2 \lambda)$
[12]	SIS	$O(\log^3 m \cdot \lambda^2)$
This work	vSIS	$O(\log^3 m \cdot \lambda^2 / \log^2 \lambda)$

Fig. 1. Asymptotic efficiency of our commitment opening proof (in bits) and comparison with prior interactive proofs which support succinct $\text{poly}[\log m, \lambda]$ verification time. Here, λ is the security parameter and m is the length of the committed vector. For each construction, the proof size corresponds to the soundness error $\text{poly}[\lambda, \log m] \cdot 2^{-\lambda}$. The SIS-related parameters were chosen with respect to the methodology from [27] for running BKZ on block size $b = O(\lambda)$. For [9, 11, 16] as well as our scheme, which only achieve inverse-polynomial soundness in one-shot, we applied a standard soundness amplification by parallel-repeating the protocol by a factor of $O(\lambda / \log \lambda)$. We highlight that for the sizes reported from [2, 12], the knowledge extractor runs in expected super-polynomial time in m and λ .

context of a specific section, we will use $\mathcal{R} \subset \mathcal{O}_{\mathcal{K}}$ to denote a ring of interest to that section. Unless specified, we measure the norm of elements and vectors by their ℓ_2 -norm over the canonical embedding over \mathcal{K} . Our results can be divided into three parts, which we overview in Sects. 2.1, 2.2, and 2.3 respectively.

2.1 Subtractive Sets

In Sect. 4, we expose subtractive sets over $\mathcal{O}_{\mathcal{K}}$ with non-prime-power conductor \mathfrak{f} , and over $\mathcal{O}_{\mathcal{K}^+}$ with both prime-power and non-prime-power conductors, with favourable properties, i.e. they have poly cardinality and small expansion factors. These subtractive sets can be used in any lattice-based arguments, and in particular those developed in this work.

A set $S \subset \mathcal{R}$ is said to be subtractive over \mathcal{R} if for any two distinct elements $c, c' \in S$, it holds that $c - c' \in \mathcal{R}^\times$, i.e. $c - c'$ is a unit. This concept is prevalently linked with the examination of Euclidean number fields [21] and has also found relevance in lattice-based cryptography, specifically in argument systems and secret sharing [3]. An explicit creation of an upper-bound-matching cardinality p is evident in a cyclotomic ring $\mathcal{R} = \mathcal{O}_{\mathcal{K}}$ with a prime-power conductor $\mathfrak{f} = p^k$. On the other hand, we are not aware of explicit studies of subtractive sets regarding other cyclotomic rings and their subrings.

For applications in lattice-based cryptography, the most relevant measures of the quality of a subtractive set S are its

- (i) cardinality $|S|$, which inversely affects the knowledge error of argument systems using S as a challenge set,

- (ii) “expansion factor” $\gamma = \gamma_S$, i.e. how much the norm of an element grows when multiplied with an element in S , which affects the “correctness gap” of lattice-based argument systems,
- (iii) “inverse-expansion factor” $\theta = \theta_S$, i.e. how much the norm of an element grow when multiplied with $(c - c')^{-1}$ for distinct $c, c' \in S$, which affects the “soundness gap” of lattice-based argument systems.

For $\mathcal{R} = \mathcal{O}_{\mathcal{K}}$ with prime-power conductor $f = p^k$, it is known [3, 21] that there exists a subtractive set S of cardinality p and expansion factors $\gamma, \theta \approx p$.

Our main result in this part is the exposition of the subtractive set $S := \{\zeta^i\}_{i \in [f/f_{\max}]}$ of cardinality $|S| = f/f_{\max}$ for any conductor f with at least two distinct prime factors, where f_{\max} is the largest prime-power factor of f . Notably, the expansion factor is $\gamma = 1$, i.e. the norm of an element does not grow when multiplied with an element from S , while the inverse-expansion factor $\theta \approx f$ is similar to the existing result for prime-power rings.

For completeness, we also expose related subtractive sets over $\mathcal{O}_{\mathcal{K}+}$ for both prime-power and non-prime-power conductors.

2.2 Tight Succinct Argument for Bounded Norm Satisfiability

In Sect. 5, we work with $\mathcal{R} = \mathcal{O}_{\mathcal{K}}$ or $\mathcal{O}_{\mathcal{K}+}$. We present a new lattice-based succinct argument for proving the bounded norm satisfiability of structured linear and/or inner-product relations, denoted by Ξ^{lin} and Ξ^{ip} respectively. More concretely, the argument system allows to prove knowledge of a short vector $\mathbf{w} \in \mathcal{R}^m$, with $m = d^\mu$, satisfying

- a linear relation $\mathbf{F}\mathbf{w} = \mathbf{y} \bmod q$, where $\mathbf{F} = \mathbf{F}_{\mu-1} \bullet \dots \bullet \mathbf{F}_0 \in \mathcal{R}_q^{n \times m}$ can be expressed as a row-wise tensor product of μ matrices $\mathbf{F}_i \in \mathcal{R}_q^{n \times d}$, and
- (optionally) an inner-product relation $\langle \mathbf{w}, \alpha(\mathbf{w}) \rangle \bmod q$, where α is either the identity function or the complex conjugate (specified publicly).

Our argument system consists of $O(\mu) = O(\log_d m)$ rounds and is public-coin, and can thus be made non-interactive via the Fiat-Shamir transform. The prover time is quasi-linear in the size of the statement, and both the proof size and the verifier time are polylogarithmic in the statement size. It can be instantiated with a transparent setup. For example, the rows of \mathbf{F} could contain a random commitment key of the vSIS commitment scheme [11] and evaluations of monomials at different evaluation points. This turns the vSIS commitment scheme into a polynomial commitment scheme, which can then be used to compile a PIOP into a SNARK.

Correctness and Soundness Gaps. A distinguishing feature of our argument system is that it is free of the so-called “correctness gap” and “soundness gap”.

The correctness gap refers to the phenomenon that although the prover’s witness \mathbf{w} is of norm at most β , the norm check performed by the verifier in the protocol is against a bound $\beta' \gg \beta$. Typically, e.g. in lattice-based Bulletproofs,

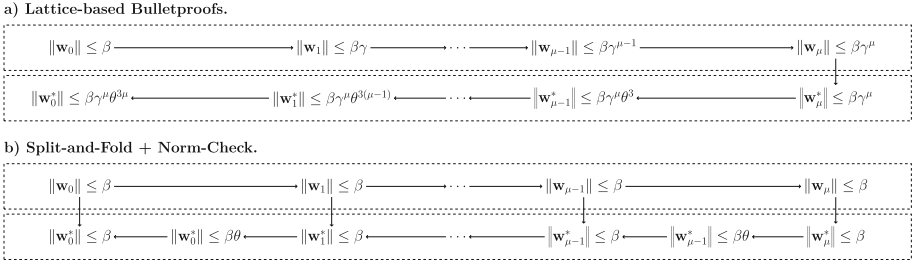


Fig. 2. Overview of the evolution of a prover witness w_0 to an extracted witness w_0^* in lattice-based Bulletproofs and in Split-and-Fold + Norm-Check.

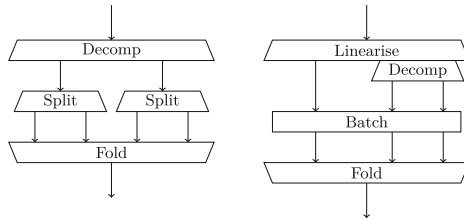


Fig. 3. Structures of Split-and-Fold (left) and Norm-Check (right) protocols.

we have $\beta' \approx (1 + \gamma)^\mu \beta$. Using the subtractive set suggested in [3] and picking $\mu \approx \log \lambda$, the gap $\beta'/\beta \approx (1 + \gamma)^\mu$ is super-polynomial in λ . Note that if the subtractive set suggested in Sect. 4 with $\gamma = 1$ is used, then the correctness gap is immediately reduced to poly but still greater than 1 (i.e. no gap).

The more challenging issue is that of the soundness gap, which refers⁵ to the limitation that, in addition to the correctness gap β'/β , the witness produced by a knowledge extractor is of even larger norm $\beta^* \gg \beta'$. Using the example of lattice-based Bulletproofs again, we have $\beta^* \approx (2\theta)^{3\mu} \beta' \approx (1 + \gamma)^\mu (2\theta)^{3\mu} \beta$. Since no currently known subtractive set (including those suggested in Sect. 4) achieves $\theta = O(1)$, the soundness gap problem cannot be solved by simply using a different subtractive set, at least until more favourable sets are found.⁶

Figure 2 overviews the evolution of a prover witness w_0 to an extracted witness w_0^* in lattice-based Bulletproofs and in this work.

Lattice-Based Bulletproofs. In Fig. 2 part a) for Bulletproofs, each arrow in the top row represents one Bulletproofs folding step, where w_i denotes the intermediate witness after the i -th folding step. The norm of the i -th round

⁵ In general, the soundness gap consists of a “stretch”, i.e. increase in witness norm, and a “slack”, i.e. a multiplicative approximation factor. Using a subtractive set, the slack can be eliminated.

⁶ We believe that a slightly better but still super-polynomial soundness gap of $\beta^*/\beta' \approx (1 + \gamma)^\mu (2\theta)^\mu$ can be achieved using a technique called “short-circuit extraction” [18].

prover witness \mathbf{w}_i grows by a multiplicative factor of (around) γ compared to the previous round prover witness \mathbf{w}_{i-1} . The last round witness \mathbf{w}_μ is then of norm around $\beta\gamma^\mu$, i.e. with correctness gap γ^μ . The vertical arrow is trivial since the last-round prover witness is sent in plain, i.e. $\mathbf{w}_\mu^* = \mathbf{w}_\mu$. Each arrow in the bottom row represents a “traditional witness extraction step”, i.e. moving one layer up in the tree-special soundness witness extraction, where \mathbf{w}_i^* denotes the extracted witness at depth i . The norm of the i -th round extracted witness \mathbf{w}_i^* grows by (roughly) a multiplicative factor of θ^3 compared to the previous round extracted witness \mathbf{w}_{i-1}^* . The final extracted witness \mathbf{w}_0^* is then of norm around $\beta\gamma^\mu\theta^{3\mu}$, i.e. the soundness gap is $\gamma^\mu\theta^{3\mu}$.

Split-and-Fold and Norm-Check. To eliminate correctness and soundness gaps, we propose two modular protocols called Split-and-Fold and Norm-Check, each of which is a composition of atomic elementary building blocks – (b -ary) Decomp, Split, Fold, Linearise, and Batch. The structures of Split-and-Fold and Norm-Check, designed to eliminate the correctness and soundness gaps respectively, are depicted in Fig. 3. These protocols are designed to run in an interleaved manner to restrict the norm growth of the witness and to aid witness extraction. The lattice-based Bulletproofs protocol can be seen as a repeated execution of the barebone Split protocol and Fold protocol without any norm-restricting mechanisms.

The Split-and-Fold and Norm-Check can be summarised as follows:

Split-and-Fold. The purpose of the Split-and-Fold protocol is to shrink the dimension of the relation to be proven without increasing the norm of the witness. On input a Ξ^{lin} instance (\mathbf{F}, \mathbf{y}) with witness \mathbf{w} of norm at most β , run the Decomp protocol to decompose the witness into b -ary parts. This splits (\mathbf{F}, \mathbf{y}) into ℓ sub-instances $(\mathbf{F}, \mathbf{y}_i)$ each with witness \mathbf{w}_i . If b is small, each sub-witness \mathbf{w}_i norms at most $b/2 \ll \beta$. Then, for each sub-instance $(\mathbf{F}, \mathbf{y}_i)$ with witness \mathbf{w}_i , run the Split protocol to peel off one tensor factor of \mathbf{F} , i.e. factor \mathbf{F} into $\mathbf{F} = \mathbf{R} \bullet \tilde{\mathbf{F}}$ and decompose \mathbf{w}_i into $(\mathbf{w}_{i,j})_{j \in [d]}$. Each sub-instance is thus further split into finer sub-instances $(\tilde{\mathbf{F}}, \mathbf{y}_{i,j})$ for some appropriate $\mathbf{y}_{i,j}$ with witnesses $\mathbf{w}_{i,j}$ still of norm at most $b/2$. Finally, the Fold protocol is run to merge all sub-instances into a single instance $(\tilde{\mathbf{F}}, \tilde{\mathbf{y}})$ with witness $\tilde{\mathbf{w}}$. For appropriately chosen b , we should end up with a next-round witness $\tilde{\mathbf{w}}$ of norm at most β again.

Note that the above suffices to eliminate the correctness gap, i.e. if the Split-and-Fold protocol were to be run recursively, the intermediate and hence final witnesses of the prover will remain to have norm at most β . However, given an extracted next-round witness $\tilde{\mathbf{w}}$ of norm at most β , the knowledge extractor could only extract a candidate witness of norm at most $\approx \theta\beta$. To improve the norm bound to β , we need to interleave a Norm-Check protocol, as described below, between executions of Split-and-Fold.

Norm-Check. The purpose of the Norm-Check protocol is to upgrade a relaxed norm bound guarantee to a tight norm bound guarantee. On input a Ξ^{lin} instance (\mathbf{F}, \mathbf{y}) with witness \mathbf{w} of norm at most β , the prover sends the value $t = \langle \mathbf{w}, \overline{\mathbf{w}} \rangle_{\mathcal{R}}$

to the verifier, who can check that $\text{Trace}(t) \leq \beta^2$. If t is computed correctly, then $\text{Trace}(t)$ is precisely the square of the canonical ℓ_2 -norm of \mathbf{w} , which the verifier checks to be at most β^2 . It thus remains for the prover to prove that t is computed correctly, along with all the other relations. To do so, the prover encodes \mathbf{w} as the coefficients of a polynomial $g(X)$, and commit to the coefficients of the Laurent polynomial $L(X) = g(X) \cdot \bar{g}(X^{-1})$. This reduces the problem to checking that L is computed correctly and has constant term t , both of which can be expressed as relations captured by Ξ^{lin} .

Two issues remain: First, the norm of the coefficients of $L(X)$ is around β^2 instead of β . To tackle this, the prover runs the Decompose protocol (in a non-black-box manner) to shrink the coefficients of $L(X)$ back to norm β , at the cost of spawning new sub-instances. Second, the extra checks for L being computed correctly and $L(0) = t$ introduce more constraints which would translate to higher communication cost when handled naively. To tackle this, the parties run the Batch protocol to compress the newly added constraints with the existing ones. Finally, we use again the Fold protocol to merge all sub-instances into one.

In Fig. 2 part b) for “Split-and-Fold + Norm-Check”, each horizontal arrow in the top row represents one “split-and-fold” step which replaces a Bulletproofs folding step. The effect of this is that the norm of \mathbf{w}_i remains at most β for all i , i.e. the correctness gap is eliminated. Each vertical arrow from \mathbf{w}_i to \mathbf{w}_i^* for $i < \mu$ represents one “Norm-Check” which is used to prove that the norm of the current witness \mathbf{w}_i is at most β . For this step, it is important that the norm function is chosen to be the ℓ_2 -norm over the canonical embedding, so that it can be expressed in terms of the (complex) inner product which is natively supported by our protocol. This proof upgrades the bound $\|\mathbf{w}_i^*\| \leq \theta\beta$ guaranteed by a traditional witness extraction step to a tighter bound $\|\mathbf{w}_i^*\| \leq \beta$, i.e. the soundness gap is eliminated.

2.3 Embedding \mathbb{Z} -Inner Products

Lattice-based succinct arguments such as those constructed in Sect. 5 typically support proving relations over a ring \mathcal{R} natively. However, in many applications, we would like to prove algebraic statements given over \mathbb{Z} , which motivates the question of how to reduce a statement over \mathbb{Z} to statements over \mathcal{R} , so that a proof of the latter implies a proof of the former. Specifically, we consider the task of proving that some (committed) vectors $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^{m\delta}$ satisfies $\langle \mathbf{x}, \mathbf{y} \rangle = z$ for some given $z \in \mathbb{Z}$. This task is of particular interest since, for some applications (e.g. constructing verifiable delay function [20]) it is necessary for the prover to prove that the witness is not only short but in fact binary. More generally, the application might require the prover to show a proof for $\mathbf{x} \in [a, b]^{m\delta}$ for some $a, b \in \mathbb{Z}$, which is not immediately implied by a bounded-norm guarantee.

To prove binariness, the basic idea is, for a witness $\mathbf{w} \in \mathbb{Z}^{m\delta}$, to use the equivalence $\mathbf{w} \in \{0, 1\}^{m\delta} \iff \langle \mathbf{1}^{m\delta} - \mathbf{w}, \mathbf{w} \rangle_{\mathbb{Z}} = 0$ to reduce checking the binariness of \mathbf{w} to checking that some transformed witness vector over \mathcal{R} is short and satisfies some linear and inner-product relations, where $\mathcal{R} \subset \mathcal{O}_{\mathcal{K}}$ is of dimension $\delta \mid \varphi$ when viewed as \mathbb{Z} -modules.

Existing Embedding Methods. We are aware of three ways to embed \mathbb{Z} -inner products into \mathcal{R} -inner products in the literature, each with a significant drawback:

- (i) Naive embedding: Interpret each \mathbb{Z} element as an \mathcal{R} element via the inclusion $\mathbb{Z} \subset \mathcal{R}$, and interpret the \mathbb{Z} -inner product as an \mathcal{R} -inner product. This incurs a multiplicative overhead of δ in terms of statement and witness sizes, which translate into overheads in prover and verifier computation, proof size, etc.
- (ii) Coefficient embedding: Divide the witness into blocks containing δ \mathbb{Z} -elements, and encode each block as an \mathcal{R} element via the (inverse-)coefficient embedding⁷ $\psi^{-1} : \mathbb{Z}^{m\delta} \rightarrow \mathcal{R}^m$. For certain \mathcal{R} , we have

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{Z}} = \text{ct}(\langle \psi^{-1}(\mathbf{x}), \overline{\psi^{-1}(\mathbf{y})} \rangle_{\mathcal{R}})$$

where $\text{ct}(\cdot)$ denotes the constant term of the coefficient embedding.

This embedding has a convenient property that it is (somewhat) norm-preserving, i.e. \mathbf{x} is short if and only if $\psi^{-1}(\mathbf{x})$ is also short (in both coefficient and canonical embedding). However, this approach only works for $\mathbb{Z}[\zeta_{2^d}]$. This is problematic since the largest subtractive set over $\mathbb{Z}[\zeta_{2^d}]$ is $\{0, 1\}$.

- (iii) CRT embedding: Let the witness vectors be such that $\mathbf{x}, \mathbf{y} \in \mathbb{Z}_p^{m\delta}$ for some (typically small) prime p which splits completely in \mathcal{R} . Divide the witness into blocks of δ \mathbb{Z} elements, and encode each block as an \mathcal{R} element via the (inverse-)CRT embedding $\text{CRT}_p^{-1} : \mathbb{Z}_p^{m\delta} \rightarrow \mathcal{R}_p^m$. It holds that

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{Z}} = \langle \mathbf{1}^\delta, \text{CRT}_p(\langle \text{CRT}_p^{-1}(\mathbf{x}), \text{CRT}_p^{-1}(\mathbf{y}) \rangle_{\mathcal{R}}) \rangle_{\mathbb{Z}} \pmod p.$$

This approach is powerful in that it not only supports proving about \mathbb{Z}_p -inner products, but in fact about \mathbb{Z}_p -Hadamard products $\mathbf{x} \odot \mathbf{y} \pmod p$, which is more fine-grained. However, to turn a claim about \mathbb{Z}_p -inner products into a claim about \mathbb{Z} -inner products (without reduction modulo p), we would additionally need to prove that $\|\langle \mathbf{x}, \mathbf{y} \rangle\|_\infty < p/2$, so that the reduction modulo p has no effect. Since CRT_p does not respect the geometry of \mathbb{Z} and \mathcal{R} , this approach usually requires the prover to commit to the witness vectors in both the $\psi^{-1}(\cdot)$ and $\text{CRT}_p^{-1}(\cdot)$ encodings, prove that the former is short, and prove that the two commitments are consistent. An issue here is that existing proofs of consistency between the two encodings (e.g. [7, 25]) do not have a succinct verifier, i.e. they run in time linear in the witness size.

In the following, we highlight how the aforementioned issues regarding the coefficient and CRT embeddings can be solved over certain (wide) range of rings.

Twisted Trace Maps. In Sect. 6, we generalise the coefficient embedding technique over power-of-2 rings to a wide range of other rings. Recall from

⁷ For example, with respect to the power basis $\{1, \zeta, \dots, \zeta^{\varphi-1}\}$ of a cyclotomic field, the coefficient embedding of an element $x = \sum_{i \in [\varphi]} x_i \zeta^i$ is denoted as $\psi(x) = (x_i)_{i \in [\varphi]}$.

the above that, over $\mathcal{O}_{\mathcal{K}}$ with a power-of-2 conductor, it holds that $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{Z}} = \text{ct}(\langle \psi^{-1}(\mathbf{x}), \overline{\psi^{-1}(\mathbf{y})} \rangle_{\mathcal{R}})$. In fact, the constant term function can be expressed as $\text{ct}(\cdot) = \frac{1}{\varphi} \cdot \text{Trace}_{\mathcal{K}/\mathbb{Q}}(\cdot)$ where $\text{Trace}_{\mathcal{K}/\mathbb{Q}}$ denotes the field trace, and the power basis $\{1, \zeta, \dots, \zeta^{\varphi-1}\}$ satisfies i.e. the power basis is orthogonal with respect to the field trace.

The above point of view motivates the search for ideal lattices with \mathbb{Z} -bases orthogonal with respect to the field trace. This leads us to the literature of lattice constellations. In particular, we extract the following embedding method from [5]: Over $\mathcal{O}_{\mathcal{K}^+}$ with prime conductor \mathfrak{f} , there exists an (efficiently computable) basis $\mathbf{b}^+ \in \mathcal{O}_{\mathcal{K}^+}^{\varphi/2}$ and a twist element $\alpha \in \mathcal{O}_{\mathcal{K}^+}$ such that

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{Z}} = \frac{1}{2\mathfrak{f}} \text{Trace}_{\mathcal{K}/\mathbb{Q}}(\alpha \cdot \langle \psi_{\mathbf{b}^+}^{-1}(\mathbf{x}), \overline{\psi_{\mathbf{b}^+}^{-1}(\mathbf{y})} \rangle_{\mathcal{R}})$$

where $\psi_{\mathbf{b}^+} : \mathcal{O}_{\mathcal{K}^+} \rightarrow \mathbb{Z}^{\varphi/2}$ denotes the coefficient embedding with respect to the basis \mathbf{b}^+ . Furthermore, adapting a result from the same work [5] regarding tensor products of rings, we extract similar embedding methods based on twisted trace maps for rings \mathcal{R} of the form $\mathcal{R} = \mathcal{O}_{\mathcal{K}_{2^d}} \otimes \mathcal{O}_{\mathcal{K}_{p_0}^+} \otimes \dots \otimes \mathcal{O}_{\mathcal{K}_{p_{k-1}}^+}$, where the subscripts of \mathcal{K} denote the conductors the respective factor rings and p_0, \dots, p_{k-1} are distinct odd primes. This captures power-of-2 rings as a special case. Notably, since such \mathcal{R} generally have non-prime-power conductors, they are compatible with the subtractive set for non-prime-power rings exposed in Sect. 4.

Succinct Proof for Consistency of CRT. As highlighted earlier, the missing piece, required to harness the power of the CRT embedding for Hadamard and inner products, is a verifier-succinct argument for proving the consistency between the coefficient embedding and the CRT embedding. More precisely, we need a succinct argument for proving that two ring vectors $\mathbf{w}, \mathbf{w}' \in \mathcal{R}^m$ satisfy

$$\psi(\mathbf{w}) = \text{CRT}_p(\mathbf{w}') \bmod p. \tag{1}$$

In the full version, we present a protocol for performing this task over $\mathcal{R} = \mathcal{O}_{\mathcal{K}}$ where the conductor \mathfrak{f} is w -smooth, i.e. all its prime factors are at most some small integer w , with proof size and verifier time scaling linearly in $w \log_w \mathfrak{f}$. In other words, if $w = O(1)$, then the complexity is logarithmic in \mathfrak{f} .

Underlying our protocol is the observation that, if the conductor \mathfrak{f} is w -smooth, then the map $\text{CRT}_p^{-1} \circ \psi$ can be expressed as the composition of $t \leq O(\log \mathfrak{f})$ maps, each being a linear combination of $h \leq O(\log \mathfrak{f})$ automorphisms from $\text{Gal}(\mathcal{K}/\mathbb{Q})$ with coefficients lying in \mathcal{R} . This means that, to succinctly prove that $\mathbf{w}' = \text{CRT}_p^{-1}(\psi(\mathbf{w})) \bmod p$, it suffices to design a succinct argument for proving automorphism relations.

Motivated by the above, we present in the full version a succinct reduction of knowledge from checking the automorphism relation $\alpha(\mathbf{w}) = \mathbf{w}'$ to checking that $(\mathbf{w}, \mathbf{w}')$ satisfies some linear relations. Combined with the Split-and-Fold and Norm-Check protocols designed in Sect. 5, we obtain a succinct argument for proving Eq. (1).

3 Preliminaries

Let $\mathbb{N} = \{1, 2, \dots\}$ denotes natural numbers and $\lambda \in \mathbb{N}$ be the security parameter. For $n \in \mathbb{N}$, we write $[n] := \{0, \dots, n - 1\}$ counting from 0. For multidimensional ranges, we use the shorthand $(i, j, k) \in [n, m, \ell]$ for $i \in [n]$, $j \in [m]$, and $k \in [\ell]$.

Throughout this work, we let $\mathcal{K} = \mathbb{Q}(\zeta)$ be a cyclotomic field with conductor f of degree $\varphi = \varphi(f)$, where ζ is a root of unity of order f and φ is Euler’s totient function, and $\mathcal{O}_{\mathcal{K}} = \mathbb{Z}[\zeta]$ be its ring of integers. We will also consider the maximal real subfield $\mathcal{K}^+ = \mathbb{Q}(\zeta + \zeta^{-1})$ of \mathcal{K} and its ring of integers $\mathcal{O}_{\mathcal{K}^+} = \mathbb{Z}[\zeta + \zeta^{-1}]$. In contexts where we refer to multiple cyclotomic fields with different conductors $(f_i)_{i \in [k]}$, we write \mathcal{K}_{f_i} for $i \in [k]$ to emphasis the conductors. We will usually use $\mathcal{R} \subseteq \mathcal{O}_{\mathcal{K}}$ to denote a subring which has dimension δ when viewed as a \mathbb{Z} -module. We assume familiarity with basic algebraic number theory.

3.1 Cryptographic Assumption

We state an equivalent formulation of the vanishing short integer solution (vSIS) assumption [11], which has a simpler description and better aligns with the notation adopted in this work.

Definition 1 (vSIS Assumption (adapted from [11])). *Let $\text{params} = (\mathcal{R}, q, \beta, \chi)$ be parametrised by λ , where \mathcal{R} is a ring, $q \in \mathbb{N}$ a modulus, $\beta > 0$ a norm bound, and χ a distribution over $\mathcal{R}_q^{n \times \otimes_{i \in [\mu]} d_i}$ for some dimensions $n, d_0, \dots, d_{\mu-1}, \mu \in \mathbb{N}$. The $\text{vSIS}_{\text{params}}$ assumption states that, for any PPT adversary \mathcal{A} , the advantage function satisfies*

$$\text{Adv}_{\text{params}, \mathcal{A}}^{\text{vSIS}} := \Pr \left[\begin{array}{l} \mathbf{F}\mathbf{w} = \mathbf{0} \pmod{q} \\ \|\sigma(\mathbf{w})\|_2 \leq \beta \end{array} \middle| \begin{array}{l} \mathbf{F} \leftarrow \mathcal{R}_q \\ \mathbf{w} \leftarrow \mathcal{A}(\mathbf{F}) \end{array} \right] \leq \text{negl}(\lambda).$$

For simplicity, in this work, we will consider the setting where the block sizes $d_0, \dots, d_{\mu-1}$ are identically set to some $d \in \mathbb{N}$, so that \mathbf{F} can be factored into $\mathbf{F} = \mathbf{F}_{\mu-1} \bullet \dots \bullet \mathbf{F}_0$ with $\mathbf{F}_i \in \mathcal{R}_q^{n \times d}$, where \bullet denotes the row-wise tensor product.

3.2 Reduction of Knowledge

In this paper we consider ternary relations $\Xi \subseteq \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^*$, where a tuple $(\text{pp}, \text{stmt}, \text{wit}) \in \Xi$ consists of public parameters pp , statement stmt and witness wit . For presentation, we omit including pp when it is known from the context. We consider a modified and simplified definition of a reduction of knowledge [19] for the following reasons: All of our protocols are *public coin* and (*coordinate-wise*) *special sound* [16] or similar.⁸ Thus, public reducibility

⁸ To turn soundness errors of probabilistic tests (such as Schwartz–Zippel) into knowledge errors, we merely need uniformly random transcripts. These are produced by (CW)SS extractors for example. We call such extractors k -transcript extractors.

is automatic and we have (super-constant) sequential composition results due to known (tree) black-box extractors, whereas composition in [19] is limited a constant number of protocols. Lastly, we define a *relaxed* knowledge soundness notion which is not present in [19]. For lack of space, we provide a condensed overview of reductions of knowledge. See the full version for details.

Definition 2 (Reduction of Knowledge (modified)). *Let Ξ_0, Ξ_1 be ternary relations. A reduction of knowledge (RoK) Π from Ξ_0 to Ξ_1 , short $\Pi: \Xi_0 \rightarrow \Xi_1$, is defined by two PPT algorithms $\Pi = (\mathcal{P}, \mathcal{V})$, the prover \mathcal{P} , and the verifier \mathcal{V} , with the following interface:*

- $\mathcal{P}(\text{pp}, \text{stmt}_1, \text{wit}_1) \rightarrow (\text{stmt}_2, \text{wit}_2)$: *Interactively reduce the input statement $(\text{pp}, \text{stmt}, \text{wit}) \in \Xi_0$ to a new statement $(\text{pp}, \widetilde{\text{stmt}}, \widetilde{\text{wit}}) \in \Xi_1$ or \perp .*
- $\mathcal{V}(\text{pp}, \text{stmt}) \rightarrow \widetilde{\text{stmt}}$: *Interactively reduce the task of checking the input statement (pp, stmt) w.r.t Ξ_0 to checking a new statement $(\text{pp}, \widetilde{\text{stmt}})$ w.r.t. Ξ_1 .*

A RoK Π is *correct*, if for any honest protocol run (with correct inputs), the prover outputs a witness for the reduced statement (which the verifier outputs). A RoK Π is *relaxed knowledge sound* from Ξ_0^{KS} to Ξ_1^{KS} with knowledge error $\kappa(\text{pp}, \text{stmt})$ if there is a *black-box* expected polynomial-time extractor \mathcal{E} , which succeeds with probability $\epsilon - \kappa(\text{pp}, \text{stmt})$ if the malicious prover outputs a valid witness for the reduced statement with probability ϵ (on verifier’s input (pp, stmt)).

4 Subtractive Sets

A subtractive set S over a ring \mathcal{R} is such that $c - c'$ is a unit for any distinct $c, c' \in S$. While the notion is connected to the study of Euclidean number fields [21], it also found applications in lattice-based cryptography in the contexts of argument systems and secret sharing [3]. For a cyclotomic ring \mathcal{R} with prime-power conductor $\mathfrak{f} = p^k$, an explicit construction of upper-bound-matching cardinality p is known. For other cyclotomic rings and their subrings, however, not much seem to be explicitly studied. In this section, we construct subtractive sets over non-prime-power cyclotomic rings. All proofs, as well as constructions of subtractive sets over *real* cyclotomic rings, can be found in the full version.

Definition 3 (Subtractive Set). *We say that a set $S \subseteq \mathcal{R}$ is subtractive over \mathcal{R} if $c - c' \in \mathcal{R}^\times$ for any distinct $c, c' \in S$.*

While [3] measured the quality of a subtractive set over cyclotomic rings in terms of the ℓ_∞ -norm over the coefficient embedding, in this work, we will instead work with the ℓ_∞ -norm over the canonical embedding for compatibility with Sect. 5 via the inequality $\forall c, x \in \mathcal{R}, \|\sigma(c \cdot x)\|_2 \leq \|\sigma(c)\|_\infty \cdot \|\sigma(x)\|_2$. We measure the quality of a subtractive set by its cardinality, expansion factor γ_S , and inverse-expansion factor θ_S , with the latter two defined below.

Definition 4 ((Inverse-)Expansion Factor of Subtractive Set). *Let $S \subseteq \mathcal{R}$ be subtractive over \mathcal{R} . The expansion and inverse-expansion factors of S are $\gamma_S := \max_{c \in S} \|\sigma(c)\|_\infty$ and $\theta_S := \max_{c, c' \in S, c \neq c'} \left\| \sigma\left(\frac{1}{c - c'}\right) \right\|_\infty$ respectively.*

4.1 Prime-Power Cyclotomics

We recall the subtractive set for prime-power cyclotomics [3, 21] with conductor $f = p^k$ and analyse its (inverse-)expansion factor in canonical ℓ_2 -norm. Although we are interested mostly in $p \gg 2$, the result also holds for $p = 2$. A proof can be found in the full version.

Theorem 1. *Let $f = p^k > 4$ for some prime p . The set $S := \{\mu_0, \dots, \mu_{p-1}\} \subseteq_p \mathcal{O}_K$ is subtractive, where $\mu_i = (\zeta^i - 1)/(\zeta - 1)$. Further, $\gamma_S \leq p$ and $\theta_S \leq \frac{f}{2\sqrt{2}}$.*

4.2 Non-prime-Power Cyclotomics

A drawback of the subtractive set recalled above is its rather large expansion factor $\gamma_S \leq p$. In some applications, e.g. Section 5, we would like γ_S to be constant. Below, we expose a subtractive set over non-prime-power cyclotomic rings with expansion factor $\gamma_S = 1$. A proof can be found in the full version.

Theorem 2. *Let f factor into $k \geq 2$ coprime prime-power factors $(\hat{f}_i)_{i \in [k]}$, i.e. $f = \prod_{i \in [k]} \hat{f}_i$. Write $\hat{f}_{\max} := \max_{i \in [k]} \hat{f}_i$. The set $S := \{1, \zeta, \zeta^2, \dots, \zeta^{f/\hat{f}_{\max}-1}\} \subseteq_{f/\hat{f}_{\max}} \mathcal{O}_K$, is subtractive. Furthermore, $\gamma_S = 1$ and $\theta_S \leq \frac{f}{4\sqrt{2}}$.*

5 Succinct Arguments for Bounded-Norm Satisfiability

In this section, we assume that \mathcal{R} is either \mathcal{O}_K or \mathcal{O}_{K^+} which admit large enough subtractive sets, e.g. those constructed in Sect. 4. Let $\mathcal{C}_{\mathcal{R}} \subset \mathcal{R}$ denote a fixed subtractive set with expansion factor γ and inverse-expansion factor θ . Throughout, we mainly use the canonical 2-norm, and simply write $\|\cdot\| := \|\sigma(\cdot)\|_2$, unless specified. We use the shorthand notation $\mathcal{R}_q^{n \times d \otimes \mu} := ((\mathcal{R}_q^{1 \times d})^{\otimes \mu})^n$ for a matrix whose rows are elementary tensors. We also write $\overline{\mathbf{Z}}$ (resp. $\underline{\mathbf{Z}}$) to indicate the top (resp. bottom) half of a block matrix; the block dimension will be clear from the context. Lastly, we let $\mathcal{C}_{\mathcal{R}_q} \subseteq \mathcal{R}_q^\times$ be obtained by taking a subfield of \mathcal{R}_q and removing 0. Note that $\mathcal{C}_{\mathcal{R}}$ and $\mathcal{C}_{\mathcal{R}_q}$ have the *invertible differences property* with respect to \mathcal{R} and \mathcal{R}_q respectively, i.e. $\forall x \neq y \in \mathcal{C}_{\mathcal{R}}$ (resp. $\mathcal{C}_{\mathcal{R}_q}$): $x - y \in \mathcal{R}^\times$ (resp. \mathcal{R}_q^\times).

We construct succinct arguments for proving that a short vector \mathbf{w} satisfy:

- \mathcal{R}_q -linear elementary tensor relations, i.e. $(\mathbf{g}_{\mu-1} \otimes \dots \otimes \mathbf{g}_0) \cdot \mathbf{w} = y \bmod q$;
- a self-inner-product relation, i.e. $t = \mathbf{w}, \alpha(\mathbf{w})_{\mathcal{R}} = \sum_{i=0}^{m-1} w_i \cdot \alpha(w)_i \in \mathcal{R}_q$; where $\alpha \in \{\text{id}, \overline{\text{id}}\}$ is either the identity map or the complex conjugate; and
- a norm bound $\|\mathbf{w}\| \leq \beta$.

Theorems and proofs are relegated to the full version. In Table 1 on page 25, we provide an overview of parameters for correctness and relaxed knowledge soundness.

5.1 The (principal) Relation Ξ^{lin}

We begin by defining the relation Ξ^{lin} and outline how protocols reduce instances in this relation to other instances. This relation serves as the principal building block for further protocols.

Basic (Single-Block) Relation. We define our central relation(s) over the ring \mathcal{R} , modulo q , for witness dimension $m = d^\mu$. In fact, there are two central relations: Ξ^{lin} for correctness; and $\Xi^{\text{lin}\vee\text{sis}}$ for relaxed knowledge soundness. We define both at once, so that $\Xi^{\text{lin}\vee\text{sis}} \supseteq \Xi^{\text{lin}}$ contains all **highlighted** parts *additionally*. Let

$$\Xi_{\mathcal{R},q,m,n^{\text{out}},\mu,\beta,\beta^{\text{sis}}}^{\text{lin}\vee\text{sis}} := \left\{ \begin{array}{l} ((\mathbf{H}, \mathbf{F}, \mathbf{y}), \mathbf{w}) : \\ \mathbf{H} \in \mathcal{R}_q^{n^{\text{out}} \times n}; \mathbf{F} \in \mathcal{R}_q^{n \times d^{\otimes \mu}} \subseteq \mathcal{R}_q^{n \times m}; \mathbf{y} \in \mathcal{R}_q^{n^{\text{out}}} \\ \left\{ \begin{array}{l} \|\mathbf{w}\| \leq \beta \\ \mathbf{H}\mathbf{F}\mathbf{w} = \mathbf{H}\mathbf{y} \pmod{q} \end{array} \right\} \text{ or } \left\{ \begin{array}{l} \|\mathbf{w}\| \leq \beta^{\text{sis}} \\ \overline{\mathbf{H}}\mathbf{F} = \mathbf{0}_{\overline{n}} \pmod{q} \end{array} \right\} \end{array} \right\}$$

where we *always assume* that \mathbf{H} has the block structure⁹

$$\mathbf{H} = \begin{pmatrix} \overline{\mathbf{H}} \\ \underline{\mathbf{H}} \end{pmatrix} \in \mathcal{R}_q^{n^{\text{out}} \times n} \quad \text{where} \quad \overline{\mathbf{H}} = (\mathbf{I}_{\overline{n}} \mathbf{0}) \in \mathcal{R}_q^{\overline{n} \times n} \quad \text{and} \quad \underline{\mathbf{H}} \in \mathcal{R}_q^{\underline{n} \times n} \quad (2)$$

Similarly, write $\overline{\mathbf{y}} \in \mathcal{R}_q^{\overline{n}}$ and $\underline{\mathbf{y}} \in \mathcal{R}_q^{\underline{n}}$ for the \overline{n} top (resp. \underline{n} bottom) rows of \mathbf{y} .

Remark 1 (Notational conventions). We often omit irrelevant parameters in Ξ^{lin} and similar relations. Especially all fixed parameters in our protocols, which are \mathcal{R} , q , \overline{n} , β^{sis} . For example, for parameterised relation like $\Xi_{\mathcal{R},q,x,y}$, we write $\Xi_{x=f(\xi)}$ for $\Xi_{\mathcal{R},q,f(\xi),z}$ or even just $\Xi_{f(\xi)}$ if $x = f(\xi)$ is clear from the context. Also, we fix d and always set $m = d^\mu$. As such, we often omit d and μ .

Clearly, relation Ξ^{lin} asserts that the witness \mathbf{w} has norm $\|\mathbf{w}\| \leq \beta$. For the linear relation, let us first assume that $\mathbf{H} = \mathbf{I}_n$ is an identity matrix. In this case, the relation asserts that $\mathbf{F}\mathbf{w} = \mathbf{y}$ holds over \mathcal{R}_q . The matrix \mathbf{F} is structured, namely each row \mathbf{f} is an elementary tensor in $\mathcal{R}_q^{1 \times d^{\otimes \mu}}$, i.e. $\mathbf{f} = \mathbf{g}_{\mu-1} \otimes \dots \otimes \mathbf{g}_0$ for $\mathbf{g}_i = (g_{i,0}, \dots, g_{i,d-1}) \in \mathcal{R}_q^{1 \times d}$.

For $\Xi^{\text{lin}\vee\text{sis}}$, we relax these assertions by introducing the **highlighted** OR-part, which captures a break of some underlying cryptographic assumption, e.g. a break of the vSIS assumption [11] (Definition 1). For this, $\overline{\mathbf{F}} = \overline{\mathbf{H}}\mathbf{F}$ will be the commitment key in a protocol. If the assumption is broken, then Ξ^{lin} may not be satisfied, hence the relaxed soundness relation $\Xi^{\text{lin}\vee\text{sis}}$ is necessary.

Now, we further explain \mathbf{H} . The primary use of \mathbf{H} is to capture *random linear combination* of rows of \mathbf{F} . The block structure asserts that the top \overline{n} rows of \mathbf{F}

⁹ This can be marginally relaxed: As long as there is an invertible $\mathbf{X} \in \mathcal{R}^{n^{\text{out}} \times n^{\text{out}}}$ such that $\mathbf{X}\mathbf{H}$ has this block structure, we can replace the claim $(\mathbf{H}, \mathbf{F}, \mathbf{y})$ with the equivalent claim $(\mathbf{X}\mathbf{H}, \mathbf{F}, \mathbf{X}\mathbf{y})$ which has the block structure our protocols require.

are simply copied— $\overline{\mathbf{F}} = \overline{\mathbf{H}}\mathbf{F}$ will correspond to the commitment key. Naively, our protocols would have communication costs linear in the number of rows of \mathbf{F} , but by using \mathbf{H} , we can compress this from n down to $n^{\text{out}} = \overline{n} + \underline{n}$. In prior works, one would simply (re)define \mathbf{F} as $\mathbf{H}\mathbf{F}$ and \mathbf{y} as $\mathbf{H}\mathbf{y}$. However, to keep (*verifier-*)*succinctness*, we cannot do this: A (random) linear combination of elementary tensors is in general not an elementary tensor. However, our protocol crucially relies on the rows of \mathbf{F} being elementary tensor in order to apply FRI-style (verifier-succinct) folding of the statement. Therefore, we remember the (random) linear combinations of rows in \mathbf{H} , instead of carrying out the multiplication. Importantly, the *communication* of the protocol can indeed be compressed by applying \mathbf{H} . (Note there that the dimensions of \mathbf{H} and \mathbf{y} are in general much smaller than that of \mathbf{w}).

Reductions between Ξ^{lin} . Our protocols reduce instances of $\Xi_{m,\beta}^{\text{lin}}$ with different parameters, and we chain them to obtain our final split-and-fold protocol with intermediate norm checks. Primary protocols and parameters of interest are:

- (i) $\Pi^{b\text{-decomp}}$: Reduce one instance with bound β to many with bound $b \ll \beta$.
- (ii) Π^{split} : Reduce one instance with witness dimension m to many with $m' = m/d$.
- (iii) Π^{fold} : Reduce many instances with bounds β_i to one with $\beta' = \gamma \sum_i \beta_i$.
- (iv) Π^{batch} : Reduce one instance to another instance by randomly combining the last \underline{n} rows of \mathbf{H} and \mathbf{y} into a single one, so that $n^{\text{out}} = \overline{n} + 1$.

Handling vSIS Breaks. Knowledge reductions can simply pass a $\Xi^{\text{lin vSIS}}$ -witness on as their extracted witness. Thus, we sometimes omit that discussion entirely.

5.2 $\Pi^{b\text{-decomp}}$: b -Ary Decomposition Knowledge Reduction

Let $b \geq 1$ be an integer. The protocol $\Pi^{b\text{-decomp}}$ (Fig. 4) is very simple: It takes a claim $((\mathbf{H}, \mathbf{F}, \mathbf{y}), \mathbf{w}) \in \Xi_{m,\beta}^{\text{lin}}$ and does a balanced b -ary decomposition of the witness \mathbf{w} with $\|\mathbf{w}\| \leq \beta$ into $\mathbf{w} = \sum_{i=0}^{\ell-1} b^i \mathbf{v}_i$, where $\mathbf{v}_i \in \mathcal{R}_b$ (hence $\|\mathbf{v}_i\|_\infty \leq b/2$) and $\ell = \lceil \log_b(2\beta + 1) \rceil$. Then, appropriate claims $\mathbf{z}_i = \mathbf{H}\mathbf{F}\mathbf{v}_i$ for the decomposed witness are computed, and the verifier makes sure the new claims imply the original one. Thus, the original statement is reduced to $((\mathbf{H}, \mathbf{F}, \mathbf{z}_i), \mathbf{v}_i)_{i \in [\ell]}$.

Remark 2. In protocol $\Pi^{b\text{-decomp}}$, we could apply the optimisation of not sending \mathbf{z}_0 , and instead let the verifier compute the unique accepting \mathbf{z}_0 , i.e. such that $\mathbf{y} = \sum_{i \in [\ell]} b^i \mathbf{z}_i$.

5.3 Π^{split} : Witness Splitting Knowledge Reduction

In Fig. 5 we describe protocol Π^{split} which takes a claim from $\Xi_{m,\beta}^{\text{lin}}$ and splits it into d claims in $\Xi_{m/d,\beta}^{\text{lin}}$. That is, the number of claims in $\Xi_{m,\beta}^{\text{lin}}$ (i.e. 1) grows by

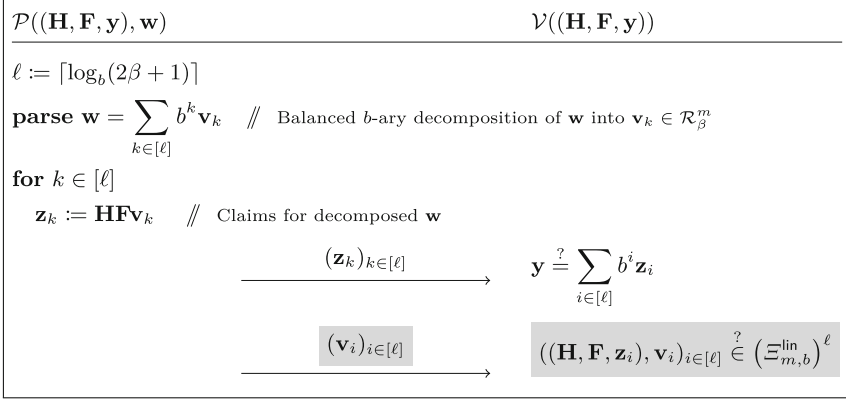


Fig. 4. Protocol $\Pi^{b\text{-decomp}}$, a reduction of $\Xi_{m,\beta}^{\text{lin}}$ to $(\Xi_{m,b}^{\text{lin}})^\ell$ for $\ell = \lceil \log_b(2\beta + 1) \rceil$. As a *proof* of knowledge, Send the **marked** parts; as a *reduction* not. $\Pi^{b\text{-decomp}}$ sends the **marked** parts only as a *proof* (but not *reduction*) of knowledge.

d -fold to d , but the witness dimension of each claim shrinks by d -fold to m/d . We explain the idea and correctness of the protocol below.

To split the witness, interpret \mathcal{R}^m as $\mathcal{R}^{d \otimes \mu}$, and split $\mathbf{w} \in \mathcal{R}^m \cong \mathcal{R}^{d \otimes \mu}$ into $\mathbf{w} = (\mathbf{w}_i)_{i \in [\mu]} = \sum_{i=0}^{\mu-1} \mathbf{e}_i \otimes \mathbf{w}_i$ where $\mathbf{w}_i \in \mathcal{R}^{m/d} \cong \mathcal{R}^{d \otimes (\mu-1)}$ and $\mathbf{e}_i \in \{0, 1\}^d$ is the i -th standard unit vector. Splitting \mathbf{w} like this is compatible with the row-wise tensor structure of \mathbf{F} . Let us take a closer look at this.

For simplicity, first consider a single row $\mathbf{f} \in \mathcal{R}_q^{1 \times d \otimes \mu}$ of \mathbf{F} . By the elementary tensor structure of the row-vector \mathbf{f} , we can write it as $\mathbf{f} = \mathbf{r} \otimes \tilde{\mathbf{f}} = (r_0 \cdot \tilde{\mathbf{f}}, \dots, r_{d-1} \cdot \tilde{\mathbf{f}}) = (\mathbf{f}_0, \dots, \mathbf{f}_{d-1})$ where $\tilde{\mathbf{f}} \in \mathcal{R}_q^{1 \times d \otimes (\mu-1)}$, $\mathbf{r} = (r_0, \dots, r_{d-1}) \in \mathcal{R}_q^{1 \times d}$, and $\mathbf{f}_i = r_i \cdot \tilde{\mathbf{f}}_i$. Therefore, $\mathbf{f} \cdot \mathbf{w} = \sum_{i \in [d]} \mathbf{f}_i \mathbf{w}_i = \sum_{i \in [d]} (\tilde{\mathbf{f}} \cdot \mathbf{w}_i) \cdot (\mathbf{r} \cdot \mathbf{e}_i^T) = \sum_{i \in [d]} r_i \tilde{\mathbf{f}} \cdot \mathbf{w}_i$.

Now, consider any matrix \mathbf{F} with row-wise tensor structure and n rows, as in Ξ^{lin} . That is, $\mathbf{F} \in \mathcal{R}_q^{n \times d \otimes \mu}$. Observe that

$$\mathbf{F} = \begin{pmatrix} \mathbf{F}_{0,\bullet} \\ \vdots \\ \mathbf{F}_{n-1,\bullet} \end{pmatrix} = \begin{pmatrix} \mathbf{F}_{0,0} & \dots & \mathbf{F}_{0,d-1} \\ \vdots & & \vdots \\ \mathbf{F}_{n-1,0} & \dots & \mathbf{F}_{n-1,d-1} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_0 \otimes \tilde{\mathbf{f}}_0 \\ \vdots \\ \mathbf{r}_{n-1} \otimes \tilde{\mathbf{f}}_{n-1} \end{pmatrix} \quad (3)$$

where $\mathbf{F}_{i,\bullet}$ denotes the i -th row of \mathbf{F} , and $\mathbf{F}_{i,j} \in \mathcal{R}_q^{1 \times d \otimes \mu}$ the block of rows (the analogue of $(\mathbf{f}_0, \dots, \mathbf{f}_{d-1})$ of the single-row case), and $\tilde{\mathbf{f}}_i \in \mathcal{R}_q^{1 \times d \otimes (\mu-1)}$ and $\mathbf{r}_i \in \mathcal{R}_q^{1 \times d}$ are the analogues of \mathbf{r} and $\tilde{\mathbf{f}}$ of the single-row case respectively. To ease notation, we define $\mathbf{R} = (\mathbf{r}_i^T)_{i \in [n]} \in \mathcal{R}_q^{n \times d}$ and $\tilde{\mathbf{F}} = (\tilde{\mathbf{f}}_i)_{i \in [n]} \in \mathcal{R}_q^{n \times d \otimes (\mu-1)}$, and we write $\mathbf{F} = \mathbf{R} \bullet \tilde{\mathbf{F}}$ for the row-wise tensor product¹⁰ of \mathbf{R} and $\tilde{\mathbf{F}}$ as seen

¹⁰ This row-wise tensor product is known under several names, e.g. row-wise Kronecker product, “face-splitting product”, “transposed Khatri–Rao product” (and more general forms, as block Kronecker product and Khatri–Rao product).

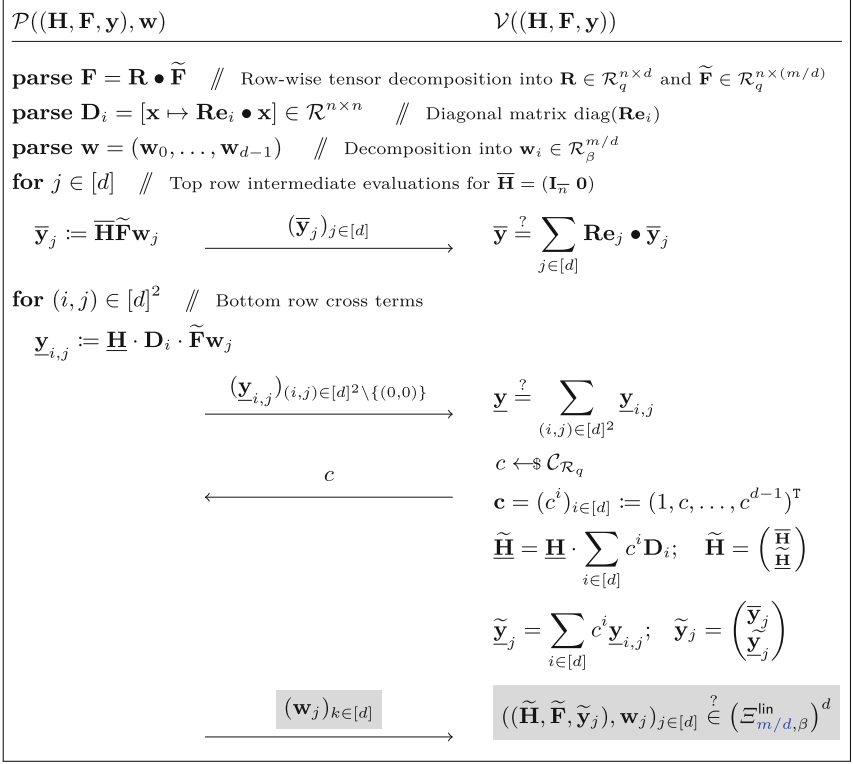


Fig. 5. Protocol Π^{split} , a reduction from $\Xi_{m, \beta}^{\text{lin}}$ to $(\Xi_{m/d, \beta}^{\text{lin}})^d$. Π^{split} sends the marked parts only as a *proof* (but not *reduction*) of knowledge.

in Eq. (3). In this notation,

$$\mathbf{F} \cdot \mathbf{w} = (\mathbf{R} \bullet \tilde{\mathbf{F}}) \cdot \left(\sum_{i=0}^{\mu-1} \mathbf{e}_i \otimes \mathbf{w}_i \right) = \sum_i \underbrace{(\mathbf{R}\mathbf{e}_i)}_{\in \mathcal{R}_q^n} \odot \underbrace{(\tilde{\mathbf{F}}\mathbf{w}_i)}_{=\mathbf{y}_i \in \mathcal{R}_q^n} \quad (4)$$

where we use the Hadamard product to multiply the vector $\mathbf{R}\mathbf{e}_i$ with \mathbf{y}_i componentwise. Moreover, with $\mathbf{D}_i := \text{diag}(\mathbf{r}_i)$, we can rewrite (4) as

$$\mathbf{F} \cdot \mathbf{w} = \sum_i \mathbf{D}_i \cdot \tilde{\mathbf{F}}\mathbf{w}_i \quad (5)$$

With the above, we have derived a splitting protocol for the special case where $\mathbf{H} = \mathbf{I}_n$ is the identity matrix: Simply send $\tilde{\mathbf{y}}_i = \tilde{\mathbf{F}}\mathbf{w}_i$ for new statements $(\mathbf{H}, \tilde{\mathbf{F}}, \tilde{\mathbf{y}}_i)$ and witnesses $(\mathbf{w}_i)_{i \in [d]}$.

When \mathbf{H} is not necessarily the identity, we must also handle the bottom part $\underline{\mathbf{H}}$ of \mathbf{H} . To do so, our protocol (cf. Fig. 5) additionally sends cross terms, namely $\underline{\mathbf{D}}_i \cdot \tilde{\mathbf{F}}\mathbf{w}_j$ for $i, j \in [d]$, which are then randomly recombined.

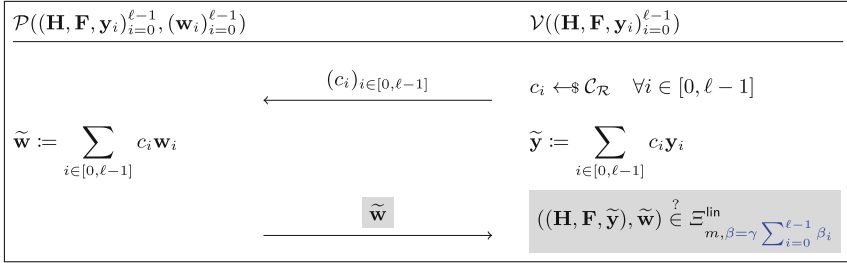


Fig. 6. Protocol Π^{fold} folds multiple instances of Ξ^{lin} with the same (\mathbf{H}, \mathbf{F}) into one. Π^{fold} sends the marked parts only as a *proof* (but not *reduction*) of knowledge.

Remark 3. In protocol Π^{split} , we could apply the optimisation of not sending $\bar{\mathbf{y}}_j$ (assuming that $\mathbf{Re}_j \neq \mathbf{0}$) and $\underline{\mathbf{y}}_{0,0}$, and instead let the verifier compute the unique accepting $\bar{\mathbf{y}}_j$ and $\underline{\mathbf{y}}_{0,0}$, i.e. such that $\bar{\mathbf{y}} = \sum_{j \in [d]} \mathbf{Re}_j \bullet \bar{\mathbf{y}}_j$ and $\underline{\mathbf{y}} = \sum_{(i,j) \in [d]^2} \underline{\mathbf{y}}_{i,j}$.

5.4 Π^{fold} : Fold Knowledge Reduction

In Fig. 6, we present the protocol Π^{fold} , which is a simple batch verification for many statements of the same type. It takes ℓ instances of $((\mathbf{H}, \mathbf{F}, \mathbf{y}_i), \mathbf{w}_i)_{i \in [\ell]}$ of $\Xi_{m, \beta}^{\text{lin}}$ with the same (\mathbf{H}, \mathbf{F}) , and produces a random linear combination $((\mathbf{H}, \mathbf{F}, \mathbf{y}), \tilde{\mathbf{w}})$ as output, with increased norm bounds.

5.5 Π^{batch} : Batch-Rows Knowledge Reduction

The protocol Π^{batch} (Fig. 7) is a protocol to batch the claims along multiple rows into fewer rows of claims. This is done by a random linear combination of the rows in question. This protocol maps an instance $((\mathbf{H}, \mathbf{F}, \mathbf{y}), \mathbf{w})$ of $\Xi_{m, \beta}^{\text{lin}}$ to an instance $((\tilde{\mathbf{H}}, \mathbf{F}, \tilde{\mathbf{y}}), \mathbf{w})$, where the dimension of $\tilde{\mathbf{y}}$ is smaller. We describe it in more detail: Let $n^{\text{out}} = \bar{n} + \underline{n}$. Then Π^{batch} keeps the top \bar{n} rows $\bar{\mathbf{y}}$ of \mathbf{y} (resp. $\bar{\mathbf{H}}$ of \mathbf{H} , and thus of \mathbf{HF}) unchanged. But the bottom \underline{n} rows are linearly combined into a single row. For this, \mathbf{H} and \mathbf{y} are split into top and bottom half, and the bottom half is multiplied by a vector \mathbf{c} consisting of powers of $c \leftarrow \mathcal{C}_{\mathcal{R}_q}$. Both parties then update the statement suitably.

5.6 $\Pi^{\text{split}\&\text{fold}}$: Split-and-Fold

We describe protocol $\Pi^{\text{split}\&\text{fold}}$ which reduces the size of the witness. While it is perfectly correct, the extracted relation suffers a (small) growth in norm, i.e. it is only relaxed knowledge sound. The protocol $\Pi^{\text{split}\&\text{fold}}$ proceeds as follows:

$$(1) \quad \Xi_{m, \beta_0}^{\text{lin}} \xrightarrow{\Pi^{b\text{-decomp}}} (\Xi_{m, b}^{\text{lin}})^\ell \quad (\text{Reduce norms by } b\text{-ary decomposition.})$$

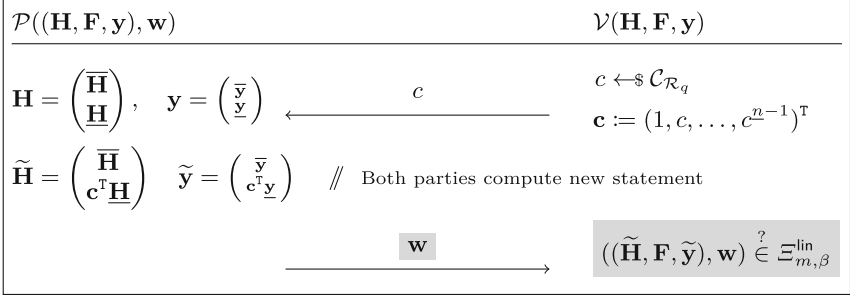


Fig. 7. Protocol Π^{batch} reduces an instance of Ξ^{lin} to another in instance with fewer rows by batching to last n of \mathbf{H} . Π^{batch} sends the marked parts only as a *proof* (but not *reduction*) of knowledge.

- (2) $(\Xi_{m, b}^{\text{lin}})^\ell \xrightarrow{\Pi^{\text{split}}} (\Xi_{m/d, b}^{\text{lin}})^{\ell d}$ (Split statements into smaller ones.)
(3) $(\Xi_{m/d, b}^{\text{lin}})^{\ell d} \xrightarrow{\Pi^{\text{fold}}} \Xi_{m/d, \beta_3}^{\text{lin}}$ (Fold statements into single one.)

The goal behind $\Pi^{\text{split\&fold}}$ is to reduce the statement size by applying Π^{split} and then Π^{fold} . However, doing just this increases the norm the folded witness. To avoid this, $\Pi^{\text{split\&fold}}$ first applies a b -ary decomposition which decreases the norm of \mathbf{w} sufficiently, with proper parameters, we get $\beta_3 = \beta_0$ again. Correctness of this protocol is straightforward to show. Relaxed knowledge soundness also follows from relaxed soundness of the building blocks. Unfortunately, the norm of the extracted witness *does* grow, no matter how the parameters are chosen.

Remark 4. When applying Π^{split} to the product relation $(\Xi_{m, b}^{\text{lin}})^\ell$, it is crucial that a single challenge is reused among all instances. This ensures that if all $(\mathbf{H}_i, \mathbf{F}_i)$ were identical before splitting, they still are identical after splitting.

5.7 Π^{norm} , $\Pi^{\text{norm+}}$, Π^{ip} , $\Pi^{\text{ip+}}$: Norm and Inner Product Checks

To restrain the norm growth of the extracted witness, we introduce norm checks. First we present the “core” norm check protocol Π^{norm} , which handles the interesting part of the norm check by reducing the norm relation Ξ^{norm} , to multiple Ξ^{lin} relations. We then compose Π^{norm} with Π^{batch} and Π^{fold} to yield the “full” norm check protocol $\Pi^{\text{norm+}}$, which reduces the norm relation to a single Ξ^{lin} relation. At the core of Π^{norm} is a mechanism for checking the trace of an inner product. By removing the trace operation, we obtain similar protocols Π^{ip} and $\Pi^{\text{ip+}}$ for proving inner product relations Ξ^{ip} . The relations Ξ^{norm} and Ξ^{ip} , as well as their variants $\Xi^{\text{norm}\vee\text{sis}}$ and $\Xi^{\text{ip}\vee\text{sis}}$, are defined as follows.

$$\Xi_{\mathcal{R}, q, m, n^{\text{out}}; \mu, \beta, \beta^{\text{sis}}}^{\text{norm}\vee\text{sis}} := \left\{ \left((\mathbf{H}, \mathbf{F}, \mathbf{y}, \nu), \mathbf{w} \right) : \begin{array}{l} \mathbf{H} \in \mathcal{R}_q^{n^{\text{out}} \times n}; \mathbf{F} \in \mathcal{R}_q^{n \times q^{\otimes \mu}} \subseteq \mathcal{R}_q^{n \times m}; \mathbf{y} \in \mathcal{R}_q^{n^{\text{out}}}, \nu \leq \beta \\ \left\{ \begin{array}{l} \|\mathbf{w}\| \leq \nu \\ \mathbf{H}\mathbf{F}\mathbf{w} = \mathbf{H}\mathbf{y} \pmod{q} \end{array} \right\} \text{ or } \left\{ \begin{array}{l} \|\mathbf{w}\| \leq \beta^{\text{sis}} \\ \overline{\mathbf{H}}\mathbf{F}\mathbf{w} = \mathbf{0}_{\overline{r}} \pmod{q} \end{array} \right\} \end{array} \right\},$$

$$\Xi_{\mathcal{R},q,m,n^{\text{out}},\mu,\beta,\beta^{\text{sis}},\alpha}^{\text{ip}\vee\text{sis}} := \left\{ \begin{array}{l} ((\mathbf{H}, \mathbf{F}, \mathbf{y}, t), \mathbf{w}) : \\ \mathbf{H} \in \mathcal{R}_q^{n^{\text{out}} \times n}; \mathbf{F} \in \mathcal{R}_q^{n \times d^{\otimes \mu}} \subseteq \mathcal{R}_q^{n \times m}; \mathbf{y} \in \mathcal{R}_q^{n^{\text{out}}}, t \in \mathcal{R}_q \\ \|\mathbf{w}\| \leq \beta \\ \left\{ \begin{array}{l} \mathbf{H}\mathbf{F}\mathbf{w} = \mathbf{H}\mathbf{y} \pmod q \\ \mathbf{w}, \alpha(\mathbf{w})_{\mathcal{R}} = t \pmod q \end{array} \right\} \text{ or } \left\{ \begin{array}{l} \|\mathbf{w}\| \leq \beta^{\text{sis}} \\ \overline{\mathbf{H}}\mathbf{F}\mathbf{w} = \mathbf{0}_{\overline{\mathcal{R}}} \pmod q \end{array} \right\} \end{array} \right\}.$$

Note that, compared to Ξ^{lin} , the norm relation Ξ^{norm} differs in that the witness norm bound $\nu \leq \beta$ is given as part of the statement, and a stricter norm relation $\|\mathbf{w}\| \leq \nu$ is checked. Similarly, Ξ^{ip} differs from Ξ^{lin} in that the statement additionally includes an inner product value t , and the witness additionally satisfies an inner product relation. Furthermore, we note that Ξ^{ip} is parametrised by $\alpha \in \{\text{id}, \overline{\text{id}}\}$ which is either the identity or complex conjugate, controlling which type of inner product is being considered.

The Core Protocols Π^{norm} and Π^{ip} . The protocols Π^{norm} , Π^{ip} for $\alpha = \text{id}$ and Π^{ip} for $\alpha = \overline{\text{id}}$ are very similar. In the following description we focus on Π^{ip} for $\alpha = \overline{\text{id}}$. Removing all conjugates yields the protocol Π^{ip} for $\alpha = \text{id}$. The protocol Π^{norm} can be obtained by letting the verifier compute the trace of the alleged inner product.

Our approach is based on polynomial identities. That is, for \mathbf{w} , we define the polynomials $g(X) = \sum_{i \in [m]} w_i X^i$ and $\bar{g}(X) = \sum_{i \in [m]} \bar{w}_i X^i$, and observe that the Laurent polynomial $L(X) = \sum_{i \in \pm[m]} v_i X^i := g(X) \cdot \bar{g}(X^{-1})$ has constant coefficient $\sum_{i \in [m]} w_i \bar{w}_i$, which is the inner product

$\mathbf{w}, \bar{\mathbf{w}}_{\mathcal{R}}$. Also observe that $v_k = \sum_{i-j=k} v_i \bar{v}_j = \overline{\text{id}} \left(\sum_{i-j=k} \bar{v}_i v_j \right) = \overline{\text{id}} \left(\sum_{j-i=k} \bar{v}_j v_i \right) = \bar{v}_{-k}$ where $v_k := 0$ if $|k| \geq m$. We exploit this symmetry to commit to $L(X)$ by committing only to (v_0, \dots, v_{m-1}) . Setting $h(X) = \sum_{i \in [m]} v_i X^i$ and $\bar{h}(X) = \sum_{i \in [m]} \bar{v}_i X^i$, we see that $L(X) = h(X) + \bar{h}(X^{-1}) - v_0$. We use this equality to prove the polynomial identity $L(X) = g(X)\bar{g}(X^{-1})$ by evaluating g, \bar{g}, h, \bar{h} at a random point $\xi \leftarrow \mathcal{C}_{\mathcal{R}_q}$ (and checking if $v_0 = t$).

However, A problem with the soundness occurs if the approach is used naively: The terms v_i have norm bounded by β^2 , so $\|\mathbf{v}\|$ which may be beyond the threshold for which the commitment is binding.

A natural approach is to run $\Pi^{b\text{-decomp}}$ to counteract this problem. However, doing so modularly runs into problems and comes at the cost of a suboptimal relaxed knowledge guarantee. We can tighten our analysis if we treat the composition with $\Pi^{b\text{-decomp}}$ as *within* the protocol Π^{ip} , i.e. we immediately send the decomposed (and binding) commitments. The reason is a technical artefact of relaxed knowledge soundness and reductions of knowledge: Relaxed soundness in $\Pi^{b\text{-decomp}}$ incurs a large factor of norm growth, however, in Π^{ip} , we do not care about the auxiliary commitment to \mathbf{v} (which norms up to $\approx \sqrt{m}\beta^2$). Thus, we can argue directly for the decomposition of \mathbf{v} into smaller \mathbf{v}_i of norm at most β , which are binding. This avoids the need for recovering recover \mathbf{v} via

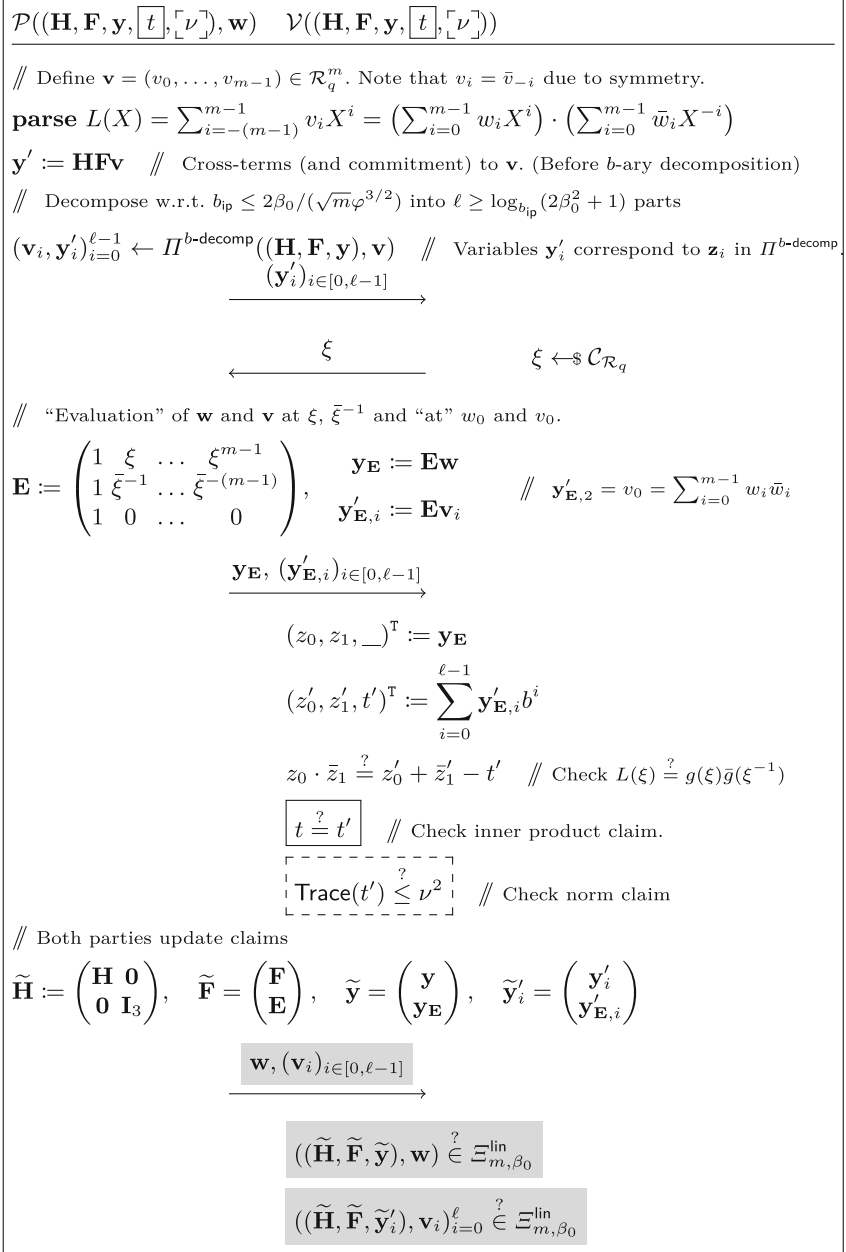


Fig. 8. Protocols $\boxed{\Pi^{\text{norm}}}$ for Ξ^{norm} and $\boxed{\Pi^{\text{ip}}}$ for Ξ^{ip} where $\alpha = \bar{\text{id}}$. The case $\alpha = \text{id}$ can be obtained by removing all conjugates. Π^{norm} and Π^{ip} send the **marked** parts only as a *proof* (but not *reduction*) of knowledge.

relaxed knowledge for $\Pi^{b\text{-decomp}}$, which significantly improves the parameters. This optimised protocol is presented in Fig. 8

The Full Protocols $\Pi^{\text{norm}+}$ and $\Pi^{\text{ip}+}$. The full protocol $\Pi^{\text{norm}+}$ (resp. $\Pi^{\text{ip}+}$) simply reduces the $\ell + 1$ statements after Π^{norm} (resp. Π^{ip}) back to a single one with the minimal number of rows, by applying Π^{batch} and Π^{fold} . That is, $\Pi^{\text{norm}+}$ (and analogously $\Pi^{\text{ip}+}$) work as follows:

- (1) $\Xi_{n^{\text{out}},\beta}^{\text{norm}} \xrightarrow{\Pi^{\text{norm}}} (\Xi_{n^{\text{out}+3,\beta}^{\text{lin}}})^{\ell+1}$ (Reduce to claims in $(\Xi^{\text{lin}})^{\ell+1}$.)
- (2) $(\Xi_{n^{\text{out}+3,\beta}^{\text{lin}}})^{\ell+1} \xrightarrow{\Pi^{\text{batch}}} (\Xi_{\bar{n}+1,\beta}^{\text{lin}})^{\ell+1}$ (Reduce number of rows.)
- (3) $(\Xi_{\bar{n}+1,\beta}^{\text{lin}})^{\ell+1} \xrightarrow{\Pi^{\text{fold}}} \Xi_{\bar{n}+1,(\ell+1)\gamma\beta}^{\text{lin}}$ (Reduce to one claim with $\beta' = (\ell + 1)\gamma\beta$.)

In words, first $\Pi^{\text{norm}+}$ (resp. $\Pi^{\text{ip}+}$) runs Π^{norm} (resp. Π^{ip}) to obtain the commitments to \mathbf{v}_i . Next it runs Π^{batch} to reduce to $\bar{n} + 1$ rows again (i.e. to $\underline{n} = 1$). Finally, the claims are folded into one by using Π^{fold} . The full norm protocol Π^{norm} and its security guarantees are analogous.

Remark 5. When applying Π^{batch} to the product relation $(\Xi_{n^{\text{out}+3,\beta}^{\text{lin}}})^{\ell+1}$, it is crucial a single challenge is reused among all instances. This ensures that if all $(\mathbf{H}_i, \mathbf{F}_i)$ were identical before batching, they still are identical after batching.

Norm Checks to Upgrade Relaxed Soundness. Currently, our norm check is defined for the relation Ξ^{norm} , which inherits the parameter β from Ξ^{lin} , and also contains an explicit ν as a norm *statement*. For convenience, we now define the $\Pi^{\text{norm}\beta}$ protocol, which is a reduction of knowledge from Ξ_{β}^{lin} to Ξ_{β}^{lin} , that works as follows: $\Pi^{\text{norm}\beta}$ runs Π^{norm} on (implicit) input $\nu = \beta$ for both parties.

Analogous to $\Pi^{\text{norm}+}$, we define $\Pi^{\text{norm}\beta+} : \Xi_{\beta}^{\text{lin}} \rightarrow \Xi_{\beta}^{\text{lin}}$. Clearly, $\Pi^{\text{norm}\beta}$ (resp. and $\Pi^{\text{norm}\beta+}$) directly inherits all correctness and security guarantees of Π^{norm} (resp. $\Pi^{\text{norm}+}$). We introduce $\Pi^{\text{norm}\beta}$ and $\Pi^{\text{norm}\beta+}$ solely for compositional reasons: They start and end with a claim(s) in Ξ_{β}^{lin} . Let us stress that the $\Pi^{\text{norm}\beta}$ protocol *upgrades* the (previously relaxed) bound on the norm of \mathbf{w} to $\|\mathbf{w}\| \leq \beta$. We capture this in following corollary.

Corollary 1. *Adopt the setting of $\Pi^{\text{ip}+}$. Then $\Pi^{\text{norm}\beta+}$ is relaxed knowledge sound from $\Xi_{\beta}^{\text{lin}\vee\text{sis}}$ to $\Xi_{\beta'}^{\text{lin}\vee\text{sis}}$ if $2\beta' \leq \beta^{\text{sis}}$, with the same knowledge error as $\Pi^{\text{ip}+}$.*

5.8 Π^{sfn} : Split-and-Fold with Norm Checks

We describe our split-and-fold protocol with intermediate norm check Π^{sfn} . It first runs the norm check $\Pi^{\text{norm}\beta+}$ to upgrade the relaxed norm bound to a strict one. Then it splits-and-folds to reduce the witness size. If parameters are set correctly, then $\Pi^{\text{split}\&\text{fold}} : \Xi_{m,\beta}^{\text{lin}} \rightarrow \Xi_{m/d,\beta}^{\text{lin}}$ is reduction of knowledge with relaxed knowledge soundness $\Xi_{m,\beta}^{\text{lin}\vee\text{sis}} \rightarrow \Xi_{m/d,\beta'}^{\text{lin}\vee\text{sis}}$. Crucially, the bound β is guaranteed exactly after extraction (unless the OR-branch, i.e. a vSIS break is extracted).

Table 1. Parameters of protocols. Expressed as $\beta_1 = f(\beta_0)$ for correctness, $\beta'_0 = g(\beta'_1)$ and $\beta'_0 \leq \beta^{\text{sis}}$ for relaxed soundness, knowledge error κ , number of transcripts to extract, other variables. For $\Pi^{\text{ip}+}$, we have $\ell = \log_{b_{\text{ip}}}(2\beta_0^2 + 1)$ for $b_{\text{ip}} \leq 2\beta/(\sqrt{m}\varphi^{3/2})$.

Π	β_1	β'_0	κ	#tr	Other
$\Pi^{b\text{-decomp}}$	$\frac{1}{2}\sqrt{m}\varphi^{3/2}b$	$2\beta_0\beta'_1$	0	1	$\ell = \lceil \log_b(2\beta_0 + 1) \rceil$
Π^{split}	β_0	$\sqrt{d}\beta'_1$	$(d-1)/ \mathcal{C}_{\mathcal{R}} $	d	Need $2\beta'_0 \leq \beta^{\text{sis}}$
Π^{fold}	$\gamma\ell\beta_0$	$2\theta\beta'_1$	$\ell/ \mathcal{C}_{\mathcal{R}} $	$\ell+1$	ℓ from $(\Xi^{\text{lin}})^{\ell}$
Π^{batch}	β_0	β'_1	$n/ \mathcal{C}_{\mathcal{R}_q} $	2	Need $2\beta'_0 \leq \beta^{\text{sis}}$
$\Pi^{\text{norm}}/\Pi^{\text{ip}}$	β_0	β'_1	$2m/ \mathcal{C}_{\mathcal{R}_q} $	2	
$\Pi^{\text{split}\&\text{fold}}$	$\frac{1}{2}\gamma\ell d\sqrt{m}\varphi^{3/2}b\beta_0$	$4\sqrt{d}\theta\beta_0\beta'_1$	$\ell/ \mathcal{C}_{\mathcal{R}} + (d-1)/ \mathcal{C}_{\mathcal{R}_q} $	$d(\ell+1)$	$\ell = \ell_{\Pi^{b\text{-decomp}}}$
$\Pi^{\text{norm}+}/\Pi^{\text{ip}+}$	$\gamma(\ell+1)\beta_0$	$8\theta\beta'_1$	$\frac{\ell+1}{ \mathcal{C}_{\mathcal{R}} } + \frac{4(\ell+1)+2m}{ \mathcal{C}_{\mathcal{R}_q} }$	$4(\ell+2)$	See caption.

- (1) $\Xi^{\text{lin}}_{\beta_0} \xrightarrow{\Pi^{\text{norm}+}_{\beta}} \Xi^{\text{lin}}_{\beta_1}$: Run a norm check for β_0 .
- (2) $\Xi^{\text{lin}}_{m,\beta_1} \xrightarrow{\Pi^{\text{split}\&\text{fold}}} \Xi^{\text{lin}}_{m/d,\beta_2}$: Run the split-and-fold to reduce witness size.

5.9 Asymptotic Communication Complexity

We now compute the proof size for the split-and-fold with norm checks protocol in Sect. 5.8. Having non-interactive proof systems in mind, we only count prover messages. Let $m = d^{\mu}$ where $d = O(1)$ and $\mu = O(\log m)$. The other parameters are chosen according to Table 1. As shown in Sect. 4, we can pick $f = \text{poly}$ and a subtractive set $\mathcal{C}_{\mathcal{R}}$ such that $\gamma = O(1)$ and $\theta = \text{poly}$.

Recall the prover executes two sub-protocols:

- (1) $\Xi^{\text{lin}}_{\beta_0} \xrightarrow{\Pi^{\text{norm}+}_{\beta}} \Xi^{\text{lin}}_{\beta_1}$: Run a norm check for β_0 .
- (2) $\Xi^{\text{lin}}_{m,\beta_1} \xrightarrow{\Pi^{\text{split}\&\text{fold}}} \Xi^{\text{lin}}_{m/d,\beta_2}$: Run the split-and-fold to reduce witness size.

First, we turn to the $\Pi^{\text{norm}+}_{\beta}$ protocol in Fig. 8. To prove relation $\Xi^{\text{lin}}_{\beta_0}$, the prover starts with

$$\Xi^{\text{norm}}_{n^{\text{out}},\beta} \xrightarrow{\Pi^{\text{norm}}} (\Xi^{\text{lin}}_{n^{\text{out}+3,\beta}})^{\ell+1}$$

where it sends all \mathbf{y}'_i which in total have size $\ell_0 n^{\text{out}}$ ring elements. After receiving the challenge ξ , the prover outputs \mathbf{y}_E and $\mathbf{y}'_{E,i}$ of size 3 each – thus in total $3(\ell_0 + 1)$ ring elements. Then the prover runs

$$(\Xi^{\text{lin}}_{n^{\text{out}+3,\beta}})^{\ell+1} \xrightarrow{\Pi^{\text{batch}}} (\Xi^{\text{lin}}_{n+1,\beta})^{\ell+1} \xrightarrow{\Pi^{\text{fold}}} \Xi^{\text{lin}}_{n+1,(\ell+1)\gamma\beta}$$

where neither Π^{batch} nor Π^{fold} incur any no communication from the prover.

Next, we now move on to the $\Pi^{\text{split}\&\text{fold}}$ protocol. The prover wants to give a proof for relation $\Xi^{\text{lin}}_{m,\beta_1}$. The prover starts by running the $\Pi^{b\text{-decomp}}$ protocol

$$\Xi^{\text{lin}}_{m,\beta_1} \xrightarrow{\Pi^{b\text{-decomp}}} (\Xi^{\text{lin}}_{m,b_1})^{\ell_1}$$

where the prover sends ℓ_1 vectors (\mathbf{z}_k) of size n^{out} elements in \mathcal{R}_q . Next, it runs Π^{split}

$$(\Xi_{m,b_1}^{\text{lin}})^{\ell_1} \xrightarrow{\Pi^{\text{split}}} (\Xi_{m/d,b_1}^{\text{lin}})^{\ell_1 d}$$

and outputs (using the optimization in Remark 4): (i) $d - 1$ intermediate “top-part” evaluations $\bar{\mathbf{y}}_j$ of size \bar{n} elements in \mathcal{R}_q and (ii) $d^2 - 1$ “bottom-part” evaluations $\underline{\mathbf{y}}_{i,j}$ of size \underline{n} elements in \mathcal{R}_q . Finally the prover executes the Π^{fold} protocol

$$(\Xi_{m/d,b_1}^{\text{lin}})^{\ell_1 d} \xrightarrow{\Pi^{\text{fold}}} \Xi_{m/d,\beta_2}^{\text{lin}}$$

where no prover message is sent. All in all, in each of $\mu = O(\log m)$ iterations of split-and-fold with norm checks, the prover sends

$$(\ell_0 n^{\text{out}} + 3(\ell_0 + 1)) + (\ell_1 n^{\text{out}} + (d - 1)\bar{n} + (d^2 - 1)\underline{n})$$

elements in \mathcal{R}_q .

Simple Example: vSIS Opening Proof. When proving knowledge of a vSIS commitment opening, we can set $n, n^{\text{out}} \in O(1)$. Due to the polynomial challenge space for subtractive sets, we need to repeat $O(\lambda/\log \lambda)$ times to ensure $\approx 2^{-\lambda}$ soundness error. Finally, we can pick b_1 such that $\ell_1 = O(1)$. In total, the proof size in the number of ring elements is simply bounded by $O\left(\lambda \frac{\log m}{\log \lambda}\right)$.

Next, we turn to setting the bound required for the (v)SIS problem to be hard. A simple calculation using the formulas from previous sections shows that we need to set

$$\beta^{\text{sis}} = \beta'_0 = 16\gamma\ell_1 d^{3/2} \sqrt{m}\varphi^{3/2} \theta^2 \beta_0 b_1 = \text{poly}[m, \lambda].$$

The next step is to estimate an asymptotic size of a ring element in \mathcal{R}_q .

Hardness of SIS. To measure hardness of vSIS, we heuristically assume that it is as hard as the plain SIS problem for the dimension $\varphi = \varphi(\mathbf{f})$. To measure hardness of SIS, we first translate the canonical norm $\|\sigma(\cdot)\|_2$ into the Euclidean norm $\|\psi(\cdot)\|_2$, and then follow the heuristic methodology from [27]. That is, let $b = O(\lambda)$ be the block size of the BKZ algorithm to find a short vector in the corresponding q -ary lattice for SIS (cf. [6]). Define the root Hermite factor as $\delta_{\text{rhf}} = \left(\frac{b(\pi b)^{1/b}}{2\pi e}\right)^{1/(2(b-1))}$. Then, SIS with matrix dimensions $\varphi \times \varphi m$ and Euclidean norm $\beta^* = \beta^{\text{sis}} \cdot \text{poly}$ is hard when $\beta^* < \min\left(2^{2\sqrt{\varphi \log q \log \delta_{\text{rhf}}}}, q\right)$. By rearranging, we get that $\varphi \log q > \log^2 \beta^*/4 \log \delta_{\text{rhf}}$. Note that

$$\log \delta_{\text{rhf}} = \frac{1}{2(b-1)} \log \left(\frac{b(\pi b)^{1/b}}{2\pi e}\right) = \Theta\left(\frac{\log b}{b}\right) = \Theta\left(\frac{\log \lambda}{\lambda}\right).$$

Finally, using the fact that $\beta^* = \text{poly}[m, \lambda]$, size of a single \mathcal{R}_q element is asymptotically

$$\Omega\left(\frac{\lambda \cdot (\log m + \log \lambda)^2}{\log \lambda}\right) = \Omega\left(\lambda \cdot \left(\frac{\log^2 m}{\log \lambda} + \log \lambda\right)\right) \text{ bits.}$$

Therefore, we deduce that the total proof size in bits is $O(\log^3 m \cdot \lambda^2/\log^2 \lambda)$.

6 Packed \mathbb{Z} -Inner Products via Twisted Trace Maps

We propose an abstract framework based on “twisted trace maps” that reduces \mathbb{Z} -inner products to \mathcal{R} -inner products over various choices of \mathcal{R} . In a nutshell, for a fixed choice of \mathcal{R} , we would like to construct a twisted trace map $\tau : \mathcal{R} \rightarrow \mathbb{Z}$ of the form shown below, where $N \in \mathbb{N}$ is some normalisation factor and $\alpha \in \mathcal{R}$ is called a “twist” element, such that the following diagram commutes:

$$\begin{array}{ccc}
 \mathbb{Z}^\delta \times \mathbb{Z}^\delta & \xrightarrow{\langle \cdot, \cdot \rangle_{\mathbb{Z}}} & \mathbb{Z} \\
 \psi^{-1}(\cdot) \times \overline{\psi^{-1}(\cdot)} \Big\downarrow & & \Big\uparrow \tau \\
 \mathcal{R} \times \mathcal{R} & \xrightarrow{\cdot_{\mathcal{R}}} & \mathcal{R}
 \end{array}
 \quad \text{where} \quad \tau : z \mapsto \frac{1}{N} \cdot \text{Trace}(\alpha \cdot z).$$

Definition 5 (Inner-Product Embedding). *Let $\mathcal{R} \subset \mathcal{O}_{\mathcal{K}}$ be a subring identified by a \mathbb{Z} -basis $\mathbf{b} \in \mathcal{R}^\delta$ of δ elements. We say that a tuple τ is an inner-product embedding over \mathcal{R} if $\tau : \mathcal{R} \rightarrow \mathbb{Z}^\delta$ is a \mathbb{Z} -linear map and, for any $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^\delta$, it holds that $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathbb{Z}} = \tau \left(\psi_{\mathbf{b}}^{-1}(\mathbf{x}) \cdot_{\mathcal{R}} \overline{\psi_{\mathbf{b}}^{-1}(\mathbf{y})} \right)$.*

The proofs of all results in this section can be found in the full version.

6.1 Power-of-Two Cyclotomics via Constant Term

As a simple concrete example, we recall a well-known folklore technique for computing the inner product over the coefficient embeddings of power-of-two cyclotomics [24].

Theorem 3. *Let $\mathcal{R} = \mathbb{Z}[\zeta_f]$ with a conductor $f = 2^k$ for some $k \in \mathbb{N}$, $\delta = \varphi = \varphi(f) = f/2$, $\tau(\cdot) = \text{ct}(\cdot) = (\psi(\cdot))_0$, where ψ denotes the coefficient embedding and $\text{ct}(\cdot)$ is the constant term of the coefficient embedding. Then τ is an inner-product embedding over \mathcal{R} .*

Remark 6. The constant term map $\text{ct}(x)$ from Theorem 3 can be expressed in terms of the Trace function as $\tau(x) = \frac{1}{\varphi} \text{Trace}(x)$, where $\varphi = f/2$ since f is a power of 2, and may be viewed as a twisted trace map $\tau(x) = \frac{1}{\varphi} \text{Trace}(\alpha \cdot x)$ with $\alpha = 1$.

As pointed out in Sect. 4, power-of-two cyclotomic rings do not admit large subtractive sets, and are therefore ill-suited for certain applications, e.g. instantiating the succinct arguments presented in Sect. 5. This motivates the search for inner-product embeddings τ over other rings.

6.2 Prime Real Cyclotomics via Twisted Trace

A natural class of rings to search for inner-product embeddings are cyclotomic rings with large prime conductors, since they admit large subtractive sets (cf. Sect. 4). Although we did not manage to design inner-product embeddings in those rings, we did so for its maximal real subring, adapting a result from lattice code theory [5, Proposition 1].

Theorem 4. Let $\mathcal{K} = \mathbb{Q}(\zeta_f)$ where f is prime and $\mathcal{R} = \mathbb{Z}[\zeta_f + \zeta_f^{-1}]$ be identified by the \mathbb{Z} -basis $\mathbf{b}^+ = \left\{ \sum_{i=[j+1]}^{\varphi/2-i} (\zeta^{\varphi/2-i} + \zeta^{-(\varphi/2-i)}) \right\}_{j \in [\varphi/2]}$. For $z \in \mathcal{R}$, let $\tau(z) = \frac{1}{2f} \text{Trace}(\alpha z)$ be a twisted trace map for the twist element $\alpha = t \cdot \bar{t}$ where $t = \zeta^{-\varphi/2} - \zeta^{\varphi/2}$. Then τ is an inner product embedding over \mathcal{R} .

The above theorem constructs inner-product embeddings for $\mathcal{R} = \mathbb{Z}[\zeta_f + \zeta_f^{-1}]$ where f is prime. This restricts the choice of \mathcal{R} quite severely, especially considering that the subtractive set constructed in Sect. 4 for $\mathbb{Z}[\zeta_f]$ or $\mathbb{Z}[\zeta_f + \zeta_f^{-1}]$ for prime f has a large expansion factor bound $\gamma_S \leq f$.

6.3 Tensor of Prime Real Cyclotomics

To allow more fine-grained parameter selection, we extend the result in Sects. 6.1 and 6.2 by constructing larger rings using the tensor product, inspired by [5, Proposition 6]. Concretely, we construct subtractive sets for rings $\mathcal{R} = \mathcal{O}_{\mathcal{K}_{2^d}} \otimes \mathcal{O}_{\mathcal{K}_{f_0}^+} \otimes \dots \otimes \mathcal{O}_{\mathcal{K}_{f_{k-1}}^+}$ for distinct odd primes f_0, \dots, f_{k-1} . Note that \mathcal{R} has conductor $\mathfrak{f} = 2^d \cdot \prod_{i \in [k]} f_i$ and degree $\delta = 2^d \cdot \prod_{i \in [k]} (f_i - 1)$. It is contained in the ring $\mathcal{O}_{\mathcal{K}_{\mathfrak{f}}}$ which admits a subtractive set S of size f/f_{\max} with expansion factor $\gamma_S = 1$ (cf. Sect. 4).

Theorem 5. Let $\mathcal{R} = \mathcal{O}_{\mathcal{K}_{\mathfrak{g}}} \otimes \mathcal{O}_{\mathcal{K}_{f_0}^+} \otimes \dots \otimes \mathcal{O}_{\mathcal{K}_{f_{k-1}}^+}$, $\mathfrak{g} = 2^d$ for some $d \in \mathbb{N}$, and f_0, \dots, f_{k-1} distinct odd primes. Let $\mathbf{b} = \mathbf{b}_{\mathfrak{g}} \otimes \left(\bigotimes_{i \in [k]} \mathbf{b}_{f_i}^+ \right)$, where $\mathbf{b}_{\mathfrak{g}}$ is the power basis for $\mathcal{R}_{\mathfrak{g}}$ and $\mathbf{b}_{f_i}^+$ is a basis for $\mathcal{R}_{f_i}^+$ defined as in Theorem 4. Then, $\tau(\cdot) = \frac{1}{t} \cdot \text{Trace}(\alpha \cdot (\cdot))$ is inner-product embedding for $\alpha = \prod_{i \in [k]} \alpha_{f_i}$, where $t = 2^k \varphi(\mathfrak{g}) \prod_{i \in [k]} f_i$.

6.4 Reducing Binariness to Bounded Norm

We show how to reduce the \mathbb{Z} -relation $\mathbf{x} \in \{0, 1\}^{m\delta}$ to an \mathcal{R} -relation natively supported by the succinct arguments presented in Sect. 5, via the inner-product embedding framework. First, we recall the following elementary fact from [24]. A proof is given in the full version.

Proposition 1. A vector $\mathbf{x} \in \mathbb{Z}^{m\delta}$ is binary if and only if $\langle \mathbf{x}, \mathbf{1}^m - \mathbf{x} \rangle_{\mathbb{Z}} = 0$.

Next, we observe the following equivalence: $\langle \mathbf{x}, \mathbf{1}^{m\delta} - \mathbf{x} \rangle_{\mathbb{Z}} = 0 \iff \langle \mathbf{x}, \mathbf{1}^{m\delta} \rangle_{\mathbb{Z}} - \langle \mathbf{x}, \mathbf{x} \rangle_{\mathbb{Z}} = 0 \iff \langle \mathbf{x}, \mathbf{1}^{m\delta} \rangle_{\mathbb{Z}} = \langle \mathbf{x}, \mathbf{x} \rangle_{\mathbb{Z}}$. This suggests the following reduction:

- (i) The prover sends two claimed values $s, t \in \mathcal{R}$ supposedly satisfying $\tau(t) = \tau(s)$
- (ii) The prover then sends a succinct proof for $\langle \psi^{-1}(\mathbf{x}), \overline{\psi^{-1}(\mathbf{1}^{\delta m})} \rangle_{\mathcal{R}} = s$ and $\langle \psi^{-1}(\mathbf{x}), \overline{\psi^{-1}(\mathbf{x})} \rangle_{\mathcal{R}} = t$.

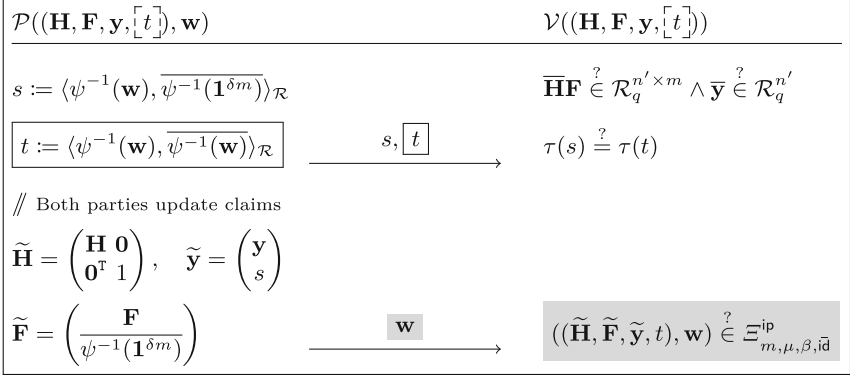


Fig. 9. Protocol $\boxed{\Pi_{\tau}^{\text{lin-bin}}}$ or $\boxed{\Pi_{\tau}^{\text{ip-bin}}}$ a reduction from $\boxed{\Xi_{m, \mu, \beta}^{\text{lin}} \cap \Xi_m^{\text{bin}}}$ or $\boxed{\Xi_{m, \mu, \beta, \text{id}}^{\text{ip}} \cap \Xi_m^{\text{bin}}}$ to $\Xi_{m, \mu, \beta, \text{id}}^{\text{ip}}$. The **marked** parts are only sent / checked when the protocol is used as a proof of knowledge. As a reduction of knowledge, they are omitted.

From the identity $\forall \mathbf{a}, \mathbf{b} \in \mathbb{Z}^{m\delta}, \tau(\langle \psi^{-1}(\mathbf{a}), \overline{\psi^{-1}(\mathbf{b})} \rangle_{\mathcal{R}}) = \langle \mathbf{a}, \mathbf{b} \rangle_{\mathbb{Z}}$, the verifier would be convinced that \mathbf{x} is indeed binary.

However, there is a subtle issue that, on one hand, the rings \mathcal{R} considered in this section are of the form displayed in Sect. 6.3, which are not necessarily equal to $\mathcal{O}_{\mathcal{K}}$ or $\mathcal{O}_{\mathcal{K}^+}$ for any cyclotomic field \mathcal{K} . On the other hand, the succinct arguments constructed in Sect. 5 are over rings which admit large subtractive sets, for which we only know constructions in $\mathcal{O}_{\mathcal{K}}$ and $\mathcal{O}_{\mathcal{K}^+}$. We therefore need to lift the \mathcal{R} -relations that we want to prove to some $\mathcal{O}_{\mathcal{K}}$ -relations (or $\mathcal{O}_{\mathcal{K}^+}$ -relations, but we focus on the former) with $\mathcal{R} \subseteq \mathcal{O}_{\mathcal{K}}$, while ensuring that the prover cannot cheat by using a witness over $\mathcal{O}_{\mathcal{K}}$. To do this, we need the lemma which allows viewing $\mathcal{O}_{\mathcal{K}}$ as an \mathcal{R} -module in such a way that the geometry of $\mathcal{O}_{\mathcal{K}}$ is respected. We refer to the full version for a precise lemma and its proof.

We next formally define the binariness relation which ignores the statement and simply checks that the witness is a binary vector.

$$\Xi_m^{\text{bin}} := \{ (\text{stmt}, \mathbf{w}) : \text{stmt} \in \{0, 1\}^*; \mathbf{w} \in \mathcal{R}^m; \psi(\mathbf{w}) \in \{0, 1\}^{m\delta} \}.$$

In Fig. 9, we present two similar reductions of knowledge $\Pi^{\text{lin-bin}}$ and $\Pi^{\text{ip-bin}}$ from $\Xi_{m, \mu, \beta}^{\text{lin}} \cap \Xi_m^{\text{bin}}$ or $\Xi_{m, \mu, \beta, \text{id}}^{\text{ip}} \cap \Xi_m^{\text{bin}}$ to $\Xi_{m, \mu, \beta, \text{id}}^{\text{ip}}$, respectively. Note that, when reducing $\Xi_{m, \mu, \beta, \text{id}}^{\text{ip}} \cap \Xi_m^{\text{bin}}$ to $\Xi_{m, \mu, \beta, \text{id}}^{\text{ip}}$, the inner product $t = \langle \psi^{-1}(\mathbf{x}), \overline{\psi^{-1}(\mathbf{x})} \rangle_{\mathcal{R}}$ is already included as part of the statement, and thus the prover does not need to send it. The formal result is stated in Theorem 6.

Theorem 6. *Let $\mathcal{R} = \mathcal{O}_{\mathcal{K}_g} \otimes \mathcal{O}_{\mathcal{K}_{p_0}^+} \otimes \dots \otimes \mathcal{O}_{\mathcal{K}_{p_{k-1}}^+}$, $g = 2^d$ for some $d \in \mathbb{N}$, and f_0, \dots, f_{k-1} distinct odd primes. Let τ be an inner-product embedding over \mathcal{R} . The protocol $\Pi_{\tau}^{\text{lin-bin}}$ (resp. $\Pi_{\tau}^{\text{ip-bin}}$) is a perfectly correct reduction of knowledge*

from $\Xi_{m,\mu,\beta}^{\text{lin}} \cap \Xi_m^{\text{bin}}$ (resp. $\Xi_{m,\mu,\beta,\text{id}}^{\text{ip}} \cap \Xi_m^{\text{bin}}$) over \mathcal{R} to $\Xi_{m,\mu,\beta,\text{id}}^{\text{ip}}$ over $\mathcal{O}_{\mathcal{K}}$. There exists a constant c_f such that it is relaxed knowledge sound from $\Xi_{m,\mu,\beta}^{\text{lin}\vee\text{sis}} \cap \Xi_m^{\text{bin}}$ (resp. $\Xi_{m,\mu,\beta,\text{id}}^{\text{ip}\vee\text{sis}} \cap \Xi_m^{\text{bin}}$) over \mathcal{R} to $\Xi_{m,\mu,\beta,\text{id}}^{\text{ip}}$ over $\mathcal{O}_{\mathcal{K}}$ if $2^k c_f \cdot \varphi^{5/2} \cdot \beta \leq \beta^{\text{sis}}$.

7 Parameter Selection

We propose concrete instantiations of our protocols for various values of m . For comparison with prior works, e.g. [2, 7], we aim for 128-bit security. This corresponds to the root Hermite factor $\delta_{\text{rhf}} \approx 1.0044$ (cf. Sect. 5.9).

Table 2. Concrete parameters, together with proof sizes, for security level $\lambda = 128$.

witness length in \mathbb{Z} -elements	$\approx 2^{18}$	$\approx 2^{20}$	$\approx 2^{24}$
$\log q$	110	110	120
f	5544	5544	5544
m	2^{20}	2^{22}	2^{26}
$\ \psi(\cdot)\ _{\infty}$ of the witness	2^5	2^5	2^5
witness size	1080 MB	4320 MB	69120 MB
d	2	2	2
μ	2	3	4
# of repetitions	17	18	19
(unoptimized) proof size	258.4 MB	263.6 MB	458.6 MB

7.1 Split-and-Fold with Norm Checks

We start with instantiating the split-and-fold with an intermediate norm check described in Sect. 5.8. We focus on the following simple goal: commit to a short vector $\mathbf{w} \in \mathbb{Z}_q^h$ of length h , such that $\|\psi(\mathbf{w})\|_{\infty} \leq \beta_{\text{init}} = 2^5$ and prove knowledge of the commitment opening. To this end, we will pack $h = m \cdot \varphi$ integers into a vector $\mathbf{w} \in \mathcal{R}_q^m$ of m ring elements employing standard coefficient embedding. Then, we will use the vSIS commitment scheme on \mathbf{w} .

The relation of our interest is a proof of vSIS commitment opening [11], i.e. the polynomial evaluation equation $\mathbf{w}(v) = y \pmod{q}$ for public ring elements $v, y \in \mathcal{R}_q$. When adapting this relation to the language of Ξ^{ip} , we would initially set $(n, n^{\text{out}}) = (1, 1)$. Throughout the batching protocols, we set $\bar{n} = 1$. In our experiments, we hardwire $f = 5544$. Hence, $\varphi = 1440$ and $|\mathcal{C}_{\mathcal{R}}| = 504$. We fix $\ell = 2$ for the whole execution of the protocol. Then, given the norm bound β and ℓ , we can deduce the decomposition base b . Since in each iteration of the split-and-fold protocol, the norm β may change, then so can the base b . We note that (at least concretely) the current proof sizes are not optimal, reaching high orders of Megabytes. We highlight that we could achieve better sizes for larger moduli.

However, conceptually this would be contradictory to the original intention of split-and-fold with norm checks; avoiding unnecessary stretch and large proof system modulus. We defer more fine-grained optimisation to a follow-up work.

Concrete Parameters. In Table 2 we suggest concrete parameters with the estimated proof size. The results are obtained via a dedicated script¹¹ simulating protocol execution and measuring the communication cost.

Acknowledgments. R.L. and M.O. are supported by the Research Council of Finland project No. 358951. This work was supported by the Helsinki Institute for Information Technology (HIIT) and conducted while M.K. was affiliated with Aalto University. N.K.N. was supported by the Protocol Labs RFP-013: Cryptonet network grant.

References

1. Albrecht, M.R., Cini, V., Lai, R.W.F., Malavolta, G., Thyagarajan, S.A.K.: Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 102–132. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_4
2. Albrecht, M.R., Fenzi, G., Lapiha, O., Nguyen, N.K.: Slap: Succinct lattice-based polynomial commitments from standard assumptions. Cryptology ePrint Archive, Paper 2023/1469 (2023), <https://eprint.iacr.org/2023/1469>, <https://eprint.iacr.org/2023/1469>
3. Albrecht, M.R., Lai, R.W.F.: Subtractive sets over cyclotomic rings - limits of Schnorr-like arguments over lattices. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part II. LNCS, vol. 12826, pp. 519–548. Springer, Heidelberg, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84245-1_18
4. Baum, C., Lyubashevsky, V.: Simple amortized proofs of shortness for linear relations over polynomial rings. Cryptology ePrint Archive, Report 2017/759 (2017), <https://eprint.iacr.org/2017/759>
5. Bayer-Fluckiger, E., Oggier, F., Viterbo, E.: New algebraic constructions of rotated $z/\text{sup } n$ -lattice constellations for the rayleigh fading channel. IEEE Transactions on Information Theory **50**(4), 702–714 (2004). <https://doi.org/10.1109/TIT.2004.825045>
6. Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. Cryptology ePrint Archive, Report 2015/1128 (2015), <https://eprint.iacr.org/2015/1128>
7. Beullens, W., Seiler, G.: LaBRADOR: Compact proofs for R1CS from module-SIS. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 518–548. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_17
8. Boneh, D., Chen, B.: Latticefold: A lattice-based folding scheme and its applications to succinct proof systems. Cryptology ePrint Archive, Paper 2024/257 (2024), <https://eprint.iacr.org/2024/257>, <https://eprint.iacr.org/2024/257>

¹¹ The script and the output are available at <https://github.com/russell-lai/rok-paper-sissors-estimator/blob/camera-ready/rok-estimator.ipynb>.

9. Bootle, J., Chiesa, A., Sotiraki, K.: Lattice-based succinct arguments for NP with polylogarithmic-time verification. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 227–251. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_8
10. Bootle, J., Lyubashevsky, V., Nguyen, N.K., Seiler, G.: A non-PCP approach to succinct quantum-safe zero-knowledge. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 441–469. Springer, Heidelberg (Aug 2020). https://doi.org/10.1007/978-3-030-56880-1_16
11. Cini, V., Lai, R.W.F., Malavolta, G.: Lattice-based succinct arguments from vanishing polynomials - (extended abstract). In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part II. LNCS, vol. 14082, pp. 72–105. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38545-2_3
12. Cini, V., Malavolta, G., Nguyen, N.K., Wee, H.: Polynomial commitments from lattices: Post-quantum security, fast verification and transparent setup. Cryptology ePrint Archive, Paper 2024/281 (2024), <https://eprint.iacr.org/2024/281>, <https://eprint.iacr.org/2024/281>
13. Debris-Alazard, T., Fallahpour, P., Stehlé, D.: Quantum oblivious lwe sampling and insecurity of standard model lattice-based snarks. Cryptology ePrint Archive, Paper 2024/030 (2024), <https://eprint.iacr.org/2024/030>, <https://eprint.iacr.org/2024/030>
14. Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: Crystals-dilithium: A lattice-based digital signature scheme. IACR Transactions on Cryptographic Hardware and Embedded Systems **2018**(1), 238–268 (Feb 2018). <https://doi.org/10.13154/tches.v2018.i1.238-268>, <https://tches.iacr.org/index.php/TCHES/article/view/839>
15. Esgin, M.F., Kuchta, V., Sakzad, A., Steinfeld, R., Zhang, Z., Sun, S., Chu, S.: Practical post-quantum few-time verifiable random function with applications to alporand. In: Borisov, N., Díaz, C. (eds.) FC 2021, Part II. LNCS, vol. 12675, pp. 560–578. Springer, Heidelberg (Mar 2021). https://doi.org/10.1007/978-3-662-64331-0_29
16. Fenzi, G., Moghaddas, H., Nguyen, N.K.: Lattice-based polynomial commitments: Towards asymptotic and concrete efficiency. Cryptology ePrint Archive, Paper 2023/846 (2023), <https://eprint.iacr.org/2023/846>, <https://eprint.iacr.org/2023/846>
17. Gentry, C., Halevi, S., Lyubashevsky, V.: Practical non-interactive publicly verifiable secret sharing with thousands of parties. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part I. LNCS, vol. 13275, pp. 458–487. Springer, Heidelberg (May / Jun 2022). https://doi.org/10.1007/978-3-031-06944-4_16
18. Hoffmann, M., Kloof, M., Rupp, A.: Efficient zero-knowledge arguments in the discrete log setting, revisited. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 2093–2110. ACM Press (Nov 2019). <https://doi.org/10.1145/3319535.3354251>
19. Kothapalli, A., Parno, B.: Algebraic reductions of knowledge. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part IV. LNCS, vol. 14084, pp. 669–701. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38551-3_21
20. Lai, R.W.F., Malavolta, G.: Lattice-based timed cryptography. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 782–804. Springer, Heidelberg (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_25
21. Lenstra, H.W.: Euclidean number fields of large degree. *Inventiones mathematicae* **38**(3), 237–254 (1976). <https://doi.org/10.1007/BF01403131>, <https://doi.org/10.1007/BF01403131>

22. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4_43
23. Lyubashevsky, V., Neven, G.: One-shot verifiable encryption from lattices. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 293–323. Springer, Heidelberg (Apr / May 2017). https://doi.org/10.1007/978-3-319-56620-7_11
24. Lyubashevsky, V., Nguyen, N.K., Plançon, M.: Lattice-based zero-knowledge proofs and applications: Shorter, simpler, and more general. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 71–101. Springer, Heidelberg (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_3
25. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Practical lattice-based zero-knowledge proofs for integer relations. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1051–1070. ACM Press (Nov 2020). <https://doi.org/10.1145/3372297.3417894>
26. Lyubashevsky, V., Nguyen, N.K., Seiler, G.: Shorter lattice-based zero-knowledge proofs via one-time commitments. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 215–241. Springer, Heidelberg (May 2021). https://doi.org/10.1007/978-3-030-75245-3_9
27. Micciancio, D., Regev, O.: Lattice-based Cryptography, pp. 147–191. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-540-88702-7_5, https://doi.org/10.1007/978-3-540-88702-7_5
28. Papamanthou, C., Shi, E., Tamassia, R., Yi, K.: Streaming authenticated data structures. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 353–370. Springer, Heidelberg (May 2013). https://doi.org/10.1007/978-3-642-38348-9_22
29. Wee, H., Wu, D.J.: Lattice-based functional commitments: Fast verification and cryptanalysis. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part V. LNCS, vol. 14442, pp. 201–235. Springer, Heidelberg (Dec 2023). https://doi.org/10.1007/978-981-99-8733-7_7



MuxProofs: Succinct Arguments for Machine Computation from Vector Lookups

Zijing Di^{1,2}(✉), Lucas Xia¹, Wilson Nguyen¹, and Nirvan Tyagi¹

¹ Stanford University, Stanford, USA
{zidi, lucasxia, wnguyen, tyagi}@stanford.edu
² EPFL, Lausanne, Switzerland

Abstract. Proofs for machine computation prove the correct execution of arbitrary programs that operate over fixed instruction sets (e.g., RISC-V, EVM, Wasm). A standard approach for proving machine computation is to prove a universal set of constraints that encode the full instruction set at each step of the program execution. This approach incurs a proving cost per execution step on the order of the total sum of instruction constraints for all of the instructions in the set, despite each step of the program only executing a single instruction. Existing proving approaches that avoid this universal cost per step (and incur only the cost of a single instruction’s constraints per step) either fail to provide zero-knowledge or rely on recursive proof composition for which security relies on the heuristic instantiation of the random oracle.

We present new protocols for proving machine execution that resolve these limitations, enabling prover efficiency on the order of only the executed instructions while achieving zero-knowledge and avoiding recursive proofs. Our core technical contribution is a new primitive that we call a succinct vector lookup argument which enables a prover to build up a machine execution “on-the-fly”. We propose succinct vector lookups for both univariate polynomial and multivariate polynomial commitments in which vectors are encoded on cosets of a multiplicative subgroup and on subcubes of the boolean hypercube, respectively. We instantiate our proofs for machine computation by integrating our vector lookups with existing efficient, succinct non-interactive proof systems for NP.

Keywords: Zero-knowledge proofs · machine computation · vector lookups

1 Introduction

Succinct non-interactive arguments of knowledge (SNARKs) [14, 39, 48, 55] enable efficient verification of NP statements. Improving prover efficiency is a key challenge in the design of SNARKs and a pathway to increase their practical deployment. In this work, we improve the prover efficiency for an important class of statements known as *machine computation* [13, 21].

In machine computation, statements are defined by the output of a program operating over a predefined fixed instruction set. A program maintains some state including an instruction pointer which determines the next instruction to execute from the instruction set. The result of an instruction execution step is an updated state and instruction pointer pointing to the next instruction to be executed.

A starting motivation for our goal of improving prover time for machine computations is another class of statements in which structure can be leveraged for prover efficiency: disjunctions [5, 32]. A disjunctive statement consists of a set of clauses, each of which is itself an NP statement. It is satisfied if there exists a satisfying witness for at least one of the clauses. Many privacy-preserving systems rely on *zero-knowledge* proofs for disjunctive statements in which the clause that is satisfied must remain hidden. A standard approach to proving the validity of a disjunctive statement in zero-knowledge is by simply encoding the clauses into a constraint system that includes the constraints for each individual clause as well as constraints for a disjunction over validity of all clauses; this constraint system over the full set of clauses is sometimes referred to as a *universal* constraint system. Any compatible zero-knowledge SNARK for NP can be used with the universal constraint system to produce a succinct proof. Unfortunately, given no shared structure between the clauses, the universal constraint system has size equal to the sum of the individual clause constraint encodings, and prover time scales accordingly. Prior work has shown how to do better in some cases [8, 9, 40, 42, 44, 46], where most recently Goel et al. show how to build SNARKs for disjunctions with NP clauses in which prover time scales with $\tilde{O}(C + \ell)$ computation where C is the constraint size of a single clause and ℓ is the number of clauses in the disjunction. This is in contrast to a $\tilde{O}(C\ell)$ cost of the universal constraint system approach.

In this work, we would like to obtain similar prover time gains by taking advantage of the similar structure found in machine computation. Machine computation resembles a disjunction as the prover would like to prove at each step that the new program state is the result of applying one of the valid instructions in the instruction set. Indeed, it is more complex than a simple disjunction as the prover needs to additionally prove that the correct instruction is executed at each step and the intermediate program state between steps is consistent; further, the prover must do this over a sequence of many execution steps. That said, the high level goal of constructing a prover that scales only with the size of executed instructions rather than the sum of executions of a universal instruction set is similar. We target a prover time of $\tilde{O}((n + \ell)C)$ versus the prover time of $\tilde{O}(n\ell C)$ achieved through universal constraint systems where n is the number of executed instructions, ℓ is the number of instructions in the instruction set, and C is the constraint size of a single instruction.

Proofs for correct execution of machine computation have received significant attention with active projects working to build proof systems for prominent instruction sets including EVM [1, 3], RISC-V [2], and WebAssembly [4]. These proofs have already been deployed to improve the scalability of blockchain

auditing, in particular, with respect to audit of smart contract execution. Now, instead of requiring auditors to execute smart contracts locally to determine and verify a new blockchain state, auditors can simply verify a succinct proof of correct machine computation for the instruction set to which the smart contract is compiled. The task of producing such a proof can be outsourced to any untrusted prover. Importantly, the proving time for producing such a proof must be manageable as it will determine the contract execution throughput that the system will be able to support. As we discuss below, deployed systems such as zkEVM [1] do not provide zero-knowledge (despite the misnomer), in part, due to prover efficiency reasons. Nevertheless, zero-knowledge is an important property for these applications and will be necessary to realize next-generation systems that support private smart contract execution [19,66].

Prior Approaches to Succinct Proofs for Machine Computation. There have been two overarching approaches to proving correct execution of machine computation. The first is through the use of *incrementally-verifiable computation* (IVC) [63] in which each instruction step is proved in sequence building on a proof for the correct execution of the program up to that point. The second approach first “unrolls” the complete program execution and proves it as a single constraint system. Figure 1 provides a summary.

Incremental Proof Systems for Machine Computation. Ben-Sasson et al. [12] demonstrate the ability to build proofs for machine execution from IVC using recursive proofs [15] in which the constraint system for each step verifies one instruction step and recursively verifies a SNARK for the previous step. This work uses a universal constraint system encoding the full instruction set at each step. This general approach can be instantiated with state-of-the-art approaches for achieving IVC [17,20,23,24,51], which avoid direct verification of recursive proofs and therefore achieve lower recursive overhead. However, this strategy will incur computation on the order of the size of the universal constraint system at each step ($\tilde{O}(\ell C)$ per step), as opposed to just the size of the executed instruction constraints ($\tilde{O}(C)$ per step).

Instead, to obviate the universal constraint system, an alternate strategy would be to commit to constraint systems for each instruction in the instruction set, e.g., in a Merkle tree commitment. At each step, the prover would open up the commitment to the instruction to be executed for the step, prove the instruction execution, and recursively verify a proof for the previous step. In concurrent work, SuperNova [50] and Protostar [22] refine this high-level blueprint building on the state-of-the-art recursion techniques [23,51] and further employing techniques in offline memory checking [16,53,60] to remove asymptotic dependence on the number of instructions in the instruction set when opening the instruction commitment. In this way, SuperNova and Protostar build proofs for machine computation using IVC that achieve $\tilde{O}((n+\ell)C)$ prover cost (formalized as *non-uniform IVC* [50]).

A drawback of all these approaches that fall under the IVC strategy is that they rely on recursively proving computations that query random oracles. For example, in each step, the prover may need to prove the execution of a recursive

verifier that queries a random oracle in its decision process. Outside of recent exploratory work [28, 29] in specialized models, we do not have secure constructions of SNARKs that prove computations which query random oracles. Thus, in practice, the security of such constructions is based on a heuristic assumption. In particular, the assumption, informally, states these constructions remain secure if the random oracle is instantiated with a particular concrete hash function. This is necessary to encode the recursive computation in a manner suitable for existing SNARKs. As such, these constructions have not been shown to be secure in the random oracle model.

As we describe next, an alternate strategy avoids IVC (and its associated heuristic assumption) by unrolling and proving the full program execution in its entirety.

Unrolled Proof Systems for Machine Computation. Unrolled proof systems for universal constraint systems incur cost on the size of the universal constraint system per instruction unrolled ($\tilde{O}(n\ell C)$) simply by repeating the universal constraint system for each execution step [11, 13, 18]. Other unroll approaches including Pantry [21], Buffet [64], and vRAM [72] avoid the use of universal constraint systems and achieve prover computation that we desire on the order only of the executed instructions ($\tilde{O}((n + \ell)C)$).

However, the unroll approach is not able to provide full zero-knowledge of program execution—at the very least, it must leak some upper bound on the number of execution steps. Prior unroll proof systems that achieve prover computation on the order of only executed instructions leak even more: Pantry [21] and Buffet [64] both require program-specific preprocessing in which the full program description must be revealed to the verifier, while vRAM [72] avoids program-specific preprocessing but reveals the number of times each instruction is executed. Indeed, deployed systems such as zkEVM [1] take an unroll approach but do not provide zero-knowledge. Intuitively, providing zero-knowledge for unrolled executions without incurring universal constraint costs is challenging; the key issue is that it cannot be known ahead of time which instruction will be executed at each execution step.

Our Approach Using Succinct Vector Lookups. In this work, we propose the first unrolled proof system for machine computation that supports zero-knowledge (beyond an upper bound on execution length) while also incurring prover computation of $\tilde{O}((n + \ell)C)$ (see below for comparison to concurrent work). In comparison to prior proof systems that are able to achieve this prover complexity: Pantry [21], Buffet [64], and vRAM [72] do not provide zero-knowledge, and SuperNova [50] and Protostar [22] rely on IVC techniques which require a heuristic instantiation of the random oracle for recursion.

To do this, our main technical contributions are new succinct proof systems for *vector lookups*. A vector lookup allows a committer to prove that a commitment to a list of vectors contains only vectors that exist in a reference table of vectors, i.e., that every vector was “looked up” from some index in the reference table. Generically, a vector lookup can be constructed from any element lookup proof system. The generic construction starts off by committing to each position

Protocols	Prover computation?	Execution leakage?	ROM-secure?
IVC w/ UC [12]	$\tilde{O}(n\ell C)$	no leakage	N
IVC w/ Exe SuperNova [50], Protostar [22]	$\tilde{O}((n + \ell)C)$	no leakage	N
Unroll w/ UC [13], Arya [18], [11], Mirage [49]	$\tilde{O}(n\ell C)$	instruction upper bound	Y
Unroll w/ Exe Pantry [21], Buffet [64]	$\tilde{O}((n + \ell)C)$	full execution	Y
vRAM [72]	$\tilde{O}((n + \ell)C)$	instruction multiplicity	Y
Sublonk [31] (Concurrent work)	$\tilde{O}(nC)$	instruction upper bound	Y
Mux-PLONK, Mux-Marlin (This work)	$\tilde{O}((n + \ell)C)$	instruction upper bound	Y
Mux-HyperPLONK (This work)	$O((n + \ell)C)$	instruction upper bound	Y

Fig. 1. Summary of strategy and characteristics of machine execution proof protocols. UC refers to using universal circuit constraints and Exe refers to using constraints just for executed instructions. The asymptotic prover time is given in terms of the number of executed instructions n , the number of instructions in the instruction set ℓ , and the constraint size of a single instruction C . Execution leakage refers to aspects of the program execution that are revealed to the verifier. The final column refers to security in the random oracle model. IVC constructions with recursive proof composition rely on a heuristic security step instead of a random oracle model security analysis. The \tilde{O} notation hides polylogarithmic factors; in particular, the prover time for Mux- HyperPLONK is strictly $O((n + \ell)C)$ and does not have polylogarithmic factors.

of the vectors in a separate commitment. Then, it homomorphically combines the commitments into a single table via a linear combination with a random verifier challenge. Finally, it runs the element lookup protocol with respect to the random table [37]. This transformation can be applied to any suitable element lookup [18, 35–37, 58, 70, 71] but results in verification that is linear in the vector size. In contrast, our new vector lookups admit succinct verification running in time logarithmic in the vector size which will be important for our machine computation application.

Equipped with our new vector lookup, we proceed to construct an unrolled proof system for machine computation. Our approach combines vector lookup arguments with proof systems for NP constraint systems that are compiled from a common information-theoretic abstraction known as a *polynomial interactive oracle proof* (polyIOP) where computation is encoded as a vector of constraints within a polynomial commitment in a preprocessing step. In our approach, the vector of constraints representing each instruction in the instruction set are encoded together in a table that represents the available instructions. Then a vector lookup is used to construct a polynomial commitment on-the-fly that represents the unrolled machine execution, “looking up” the constraints for the executed instructions.

We instantiate this approach with three existing polyIOPs for NP, PLONK [38], Marlin [30], and HyperPLONK [27]. PLONK and Marlin are both *univariate* polyIOPs: when combined with our univariate vector lookup `CosetLookup` and a suitable univariate polynomial commitment scheme (e.g., Marlin-KZG [30, 47]), the resulting proof systems for machine computation, Mux-PLONK and Mux-Marlin, admit constant proof size at the expense of quasi-

linear proving time $\tilde{O}((n + \ell)C)$. We also build on the multivariate polyIOP, HyperPLONK, which when combined with our multivariate vector lookup SubcubeLkup and a suitable multivariate polynomial commitment scheme (e.g., Brakedown [43] or Orion [27, 65]) results in Mux-HyperPLONK with an efficient linear-time ($O((n + \ell)C)$) prover and sublinear proof size.

Minimal instruction sets (e.g., TinyRAM [13] or RISC-V) have instruction set size ℓ of 30-50 instructions and instruction constraint size C on the order of 128 constraints (in the case of enforcing 64-bit computation modulo a prime). Our evaluation estimates indicate our protocols incur less than $2.5\times$ overhead on top of comparable zero-knowledge proof systems for the same computation size. Given our protocols reduce the computation size by a factor of ℓ (i.e., $30 - 50\times$) over unrolled zero-knowledge proof systems for universal constraints, our cost accounting indicates our protocols reduce proving time by up to $9\times$ in this setting.

Further, in industry, there is an ongoing trend away from minimal instruction sets towards richer “virtual instruction sets”. In this setting, our protocols demonstrate their fullest potential. Take, for example, the Ethereum virtual machine (EVM) instruction set which includes custom instructions for Keccak hashing and ECDSA signature verification. In a rich instruction set, ECDSA verification can be represented as a single instruction with size C of 1.5 million constraints¹. In contrast, compiling ECDSA verification for a minimal instruction set greatly increases the number of executed instructions n ; ECDSA verification expands to around 5 million RISC-V instructions². Due to our protocols’ succinctness and prover efficiency with respect to C and ℓ , they are especially well suited for the task of proving machine computation for rich instruction sets that include many complex instructions.

Summary of Contributions. We summarize our contributions as follows:

- *Succinct vector lookup arguments:* We present two new vector lookup arguments that provide succinct verification and proof size with respect to vector size, lookup size, and table size. One argument, CosetLkup, encodes vectors within univariate polynomial commitments over cosets of multiplicative subgroups. The other, SubcubeLkup, encodes vectors within multivariate polynomial commitments over subcubes of the boolean hypercube.
- *MuxProofs protocols for machine computation:* We demonstrate the modularity of our approach by combining our vector lookup arguments with three different polyIOPs for NP (with different prover properties) to build zero-knowledge unrolled proof systems for machine computation that only incur prover cost on the order of executed instructions (avoiding universal constraint costs per execution step).
- *Implementation and evaluation:* We implement our univariate vector lookup CosetLkup and evaluate its efficiency. Verifier time is vastly improved over the naive linear combination approach with roughly equal verifier times at vector

¹ <https://github.com/0xPARC/circom-ecdsa>.

² <https://github.com/risc0/risc0/tree/v0.16.0/examples/ecdsa>.

size 8 and $60\times$ faster for vector size 2^{10} . We further perform a thorough cost accounting and estimate performance of our proofs for machine computation. Our full evaluation is deferred to the full version of the paper [33].

- *Generic zero-knowledge compiler for univariate polyIOPs*: We formalize common techniques for adding zero-knowledge to a sound polyIOP within a generic compiler. To do this, we introduce a *domain admissibility* property for polyIOPs that restricts how polynomial oracles may depend on witness elements and we restrict polynomial evaluations on oracles to particular polynomial identity tests.

Concurrent Work. There exist a number of concurrent works targeting improvements in prover time for proving correct machine execution. The work most closely related to MuxProofs is Sublonk [31] which achieves similar asymptotic prover time and also uses succinct vector lookups (referred to as segment lookups). Sublonk builds a univariate vector lookup from the cq element lookup protocol [35] which admits a prover time independent of the table size ℓ , $\tilde{O}(nC)$, as opposed to $\tilde{O}((n+\ell)C)$ in this work. Campanelli et al. also propose a succinct vector lookup based on cq (referred to as matrix lookups) [26]. Their protocol is more concretely efficient than that of Sublonk and also includes a zero-knowledge analysis. Even so, constant factors of the cq-based vector lookup are higher than our univariate vector lookup `CosetLkup`. In the machine computation application, we typically expect the number of executed instructions in a program n to eclipse the instruction set size ℓ and so `CosetLkup` will outperform in this case. Further, Sublonk applies their vector lookup to the PLONK polyIOP for a *layered branching circuit* computation model; their treatment does not directly address certain challenges of the machine computation model such as variable-length execution (e.g., their preprocessing work is dependent on the full execution length). Lastly, both of these works only build succinct univariate vector lookups. We further propose `SubcubeLkup`, a succinct multivariate vector lookup, enabling a strictly linear-time prover. We summarize the state of vector lookups in Fig. 2.

Jolt [7] proposes the use of the Lasso lookup for structured large tables [62] to encode and lookup executions from full instruction input-output tables (e.g., for 64-bit RISC-V, tables of size 2^{128}). In contrast, MuxProofs encodes instruction constraints for looking up which instruction to execute at each step. These are orthogonal (but possibly complementary) applications of lookups for machine computation.

Another line of work achieves the same asymptotic prover efficiency (with very good constants) but does not provide succinctness; proof size and verification time grow linearly in nC [41, 68]. Most of the work described so far considers C as the upper bound of instruction constraint size for all instructions in the instruction set, thus incurring overhead when instructions are of varying sizes. Yang et al. consider “tight” machine computation in which the prover only incurs cost on the constraint size of the executed instructions [69]. Instead of leaking the upper bound on the number of executed instructions, this model leaks the upper bound on the number of executed constraints.

Protocol	Proof size	Prover computation		Verifier computation	U/M	
Plookup [37] + LC	$(5 + m)\mathbb{G}$	$9\mathbb{F}$	$O((n + \ell) \cdot m)\mathbb{G}$	$\tilde{O}((n + \ell) \cdot m)\mathbb{F}$	$2P$	$O(m)\mathbb{G}$ U
cq [35] + LC	$(8 + m)\mathbb{G}$	$3\mathbb{F}$	$O(n \cdot m)\mathbb{G}$	$\tilde{O}(n \cdot m)\mathbb{F}$	$5P$	$O(m)\mathbb{G}$ U
Segment lookup [31]	$20\mathbb{G}$	$6\mathbb{F}$	$\tilde{O}(n \cdot m)\mathbb{G}$	$\tilde{O}(n \cdot m)\mathbb{F}$	$23P$	- U
Matrix lookup [26]	$16\mathbb{G}$	$2\mathbb{F}$	$O(n \cdot m)\mathbb{G}$	$\tilde{O}(n \cdot m)\mathbb{F}$	$13P$	- U
CosetLkup (this work)	$28\mathbb{G}$	$31\mathbb{F}$	$O((n + \ell) \cdot m)\mathbb{G}$	$\tilde{O}((n + \ell) \cdot m)\mathbb{F}$	$2P$	- U
SubcubeLkup (this work)	$O(\log((n + \ell) \cdot m))\mathbb{F}$			$- O((n + \ell) \cdot m)\mathbb{F}$	$O(\log((n + \ell) \cdot m))\mathbb{F}/H$	M

Fig. 2. Comparison of properties of vector lookup proof systems for vectors of length m , tables of ℓ vectors, and lookups of n vectors. The LC annotation denotes the generic linear combination approach to transform any field lookup with linearly-homomorphic commitments into a vector lookup [37]. The final column indicates whether the lookup operates over a univariate polynomial encoding (U) or a multivariate polynomial encoding (M). Univariate protocols are instantiated with the Marlin-KZG polynomial commitment [30] over a bilinear pairing group. For cost analysis, \mathbb{G} denotes group element/multiplication, \mathbb{F} denotes scalar field element/operation, H denotes a hash operation, and P denotes a pairing operation. Our multivariate protocol is instantiated with a linear-prover polynomial commitment with a logarithmic proof size and verifier time (e.g., Orion [27, 65]). The \tilde{O} notation hides polylogarithmic factors. We highlight that SubcubeLkup does not have polylogarithmic factors in the prover time.

SuperNova [50] and Protostar [22] achieve the same asymptotic prover time as MuxProofs but take the IVC approach with heuristic security. IVC incurs prover overhead to recursively verify a folding proof: this entails using cycles of elliptic curves and non-native arithmetic constraints [52, 56]. However, even with these overheads, IVC has been shown to be more prover-efficient than monolithic (i.e., unrolled) proof systems for many settings [57]. Our multivariate protocol Mux-HyperPLONK may offer a promising alternative, as it can be instantiated with prover-efficient polynomial commitments (e.g., Brakedown [43] or Orion [27, 65]) that do not provide required homomorphism for efficient folding in SuperNova and Protostar.

2 Technical Overview

A standard approach to constructing succinct zero knowledge proof systems employs *holography* in which the claimed computation to be proved is encoded within a *computation commitment* in an initial preprocessing phase [30, 59]. After checking the validity of the computation commitment once—a non-succinct operation that can take time linear in the size of the computation—the verifier can verify any number of proofs for the computation succinctly. Unfortunately in machine execution, the description of the unrolled executed computation of a program (i.e., the sequence of executed instructions) is dependent on the program and program inputs. Thus, a different computation commitment and verifier check would be required for each different program execution. Not only does this approach not result in succinct verification but it is also not amenable

to zero-knowledge: the executed computation description may leak information about the program and its inputs.

We describe below an overview of our strategy for constructing the first zero-knowledge argument for unrolled machine execution with prover-efficiency on the order of the executed instructions that avoids recursive proving techniques. We describe two main technical contributions (Sects. 2.2 and 2.3, respectively). The first contribution is a new building block, a succinct vector lookup argument, for efficiently proving correspondence of vector encodings between two polynomials. The second contribution is to show how to compose vector lookup arguments with holographic polyIOPs to realize succinct and prover-efficient proofs for machine computation.

2.1 Strategy: Computation Commitments from Machine Commitments

Despite the executed computation being program-dependent, there exists structure in the computation that we can take advantage of. Namely, the set of possible instructions that can be executed is fixed ahead of time as a description of the “machine” the program runs on (e.g., a RISC-V CPU has a fixed instruction set). In our work, during preprocessing, the machine description (i.e., instruction set) is encoded within a *machine commitment*. To prove machine execution of a program, a computation commitment for the executed computation (i.e., the particular sequence of executed instructions) can be computed on-the-fly in such a way that the verifier can succinctly verify correctness of the computation commitment given the machine commitment. Then given the computation commitment, we can largely rely on previous techniques to verify correctness of computation execution. As we describe next, the core insight of our work is a new way to encode computation descriptions to enable efficient proofs for the relation between the executed computation commitment and machine commitment.

Modeling Machine Execution. First, we provide an introduction to our model of machine execution. We say a machine description consists of ℓ instructions each of which are represented as a computation over an input state $(inst_{in}, mem_{in})$ and produce an output state $(inst_{out}, mem_{out})$ ³. The output state $(inst_{out}, mem_{out})$ is passed as input to the next instruction. There are two parts to the running state. First, the instruction pointer $inst \in [\ell]$ specifies which of the ℓ instructions to run next. We assume that the instruction computation checks that the instruction pointer in the input state is correct. Second, the memory mem contains all other state including program inputs, program description, program counter, and external memory. As such, in applying an instruction computation to move from $(inst_{in}, mem_{in})$ to $(inst_{out}, mem_{out})$, our modeling of

³ There are different models for computation. For example, if modeled using circuit satisfiability, an instruction circuit would take in $(inst_{in}, mem_{in}, inst_{out}, mem_{out})$ as well as possibly some other witness inputs such that the circuit is satisfied if and only if $(inst_{out}, mem_{out})$ is a valid application of the instruction computation to $(inst_{in}, mem_{in})$.

an instruction computation captures two possibly distinct functionality: (1) The instruction functionality applying changes to external memory (e.g., storing the sum of two values in the case of an “add” instruction), and (2) the control logic functionality determining the next instruction to run (e.g., changing the program counter according to inputs and reading the program description to determine the next instruction pointer).

In practice, it will not be desirable to pass the full memory as described into each instruction computation. *Offline memory checking* techniques [16] enable a verifier to efficiently check a prover maintains memory correctly by performing a small amount of work per memory access (e.g., a Merkle path check or a multiset hash update) [41, 49, 53, 60, 67]. Offline memory checking is not a contribution of this work, and any of these existing techniques can be employed with MuxProofs. In the remainder of the paper, we will consider *mem* as a small digest (e.g., a Merkle root or a multiset hash) and appropriate witnesses are provided for checking memory accesses within the instruction computation.

Encoding a Machine Commitment as a Polynomial. We now return to our goal of encoding a machine description as a machine commitment in a useful manner. Among prior proof systems that employ holography [27, 30, 38], a predominant approach to encoding the computation is as a vector (or small number of vectors) containing elements of a field \mathbb{F} . The vector, say of length m , is then encoded as the evaluation points of a polynomial over some specified domain. Some proof systems encode as a univariate polynomial $f \in \mathbb{F}^{\leq m}[X]$ of degree m where f is interpolated over evaluations of a canonical ordered subgroup $\mathbb{H} \subseteq \mathbb{F}$ [30, 38]. Others encode as a $(\log m)$ -multivariate polynomial $f \in \mathbb{F}[X_{\lceil \log m \rceil}]$ where f is the multilinear polynomial interpolated over evaluations of the boolean hypercube $\{0, 1\}^{\log m}$ [27, 59]. This preprocessing of computation commitments as polynomials is used by a popular class of proof systems known as *polynomial interactive oracle proofs* (polyIOPs) [25]. We will model our machine commitment in the same way. For now, let us focus on the univariate polynomial setting; we will revisit the multivariate polynomial setting which admits various tradeoffs shortly.

Say each instruction can be described by a vector of field elements of size m . By packing the instruction vectors into a larger vector of size ℓm , we can encode the full instruction set into the evaluations of a polynomial t over a subgroup \mathbb{H} of size $|\mathbb{H}| = \ell m$. Looking forward, a key insight to enable our efficient proof techniques is the manner in which we perform this encoding. In particular, we encode each instruction over a size- m coset of \mathbb{H} that admits useful structure. This will allow us to prove more granular properties at the level of certain instructions rather than being limited to simply proving properties about the full instruction set. More precisely, say $\mathbb{H} = \langle \omega \rangle$ is generated by generator $\omega \in \mathbb{F}$: $\mathbb{H} = \{1, \omega, \omega^2, \dots, \omega^{\ell m - 1}\}$. Then, we define a multiplicative subgroup $\mathbb{V} \leq \mathbb{H}$ of size $|\mathbb{V}| = m$ where \mathbb{V} is generated by $\gamma = \omega^\ell$:

$$\mathbb{V} = \left\{ 1, \gamma = \omega^\ell, \gamma^2 = \omega^{2\ell}, \dots, \gamma^{m-1} = \omega^{(m-1)\ell} \right\}.$$

Further, we define the ℓ cosets of \mathbb{V} in \mathbb{H} as

$$\forall i \in [0, \ell), \quad \omega^i \mathbb{V} = \left\{ \omega^i, \omega^i \gamma = \omega^{\ell+i}, \omega^i \gamma^2 = \omega^{2\ell+i}, \dots, \omega^i \gamma^{m-1} = \omega^{(m-1)\ell+i} \right\}.$$

In this way, we interpolate polynomial t for the machine commitment such that the evaluations on coset $\omega^i \mathbb{V}$ are set to the vector of field elements that describe the computation for the i^{th} instruction.

Building an Executed Computation Commitment via a Lookup Argument. Now given a polynomial t that encodes the set of ℓ instructions as a machine commitment, our goal is to produce a computation commitment polynomial for the unrolled execution. An unrolled execution consists of applying n instruction computations in sequence where n is the number of execution steps until program termination.

At a high level, we want to be able to produce a polynomial f interpolated over a subgroup \mathbb{G} (where $|\mathbb{G}| = mn$ and generator $\mathbb{G} = \langle \mu \rangle$) that encodes the n executed instructions. Analogous to our encoding of ℓ instructions from the instruction set in the machine commitment polynomial t , we encode the n executed instructions in f as evaluations over the n cosets of \mathbb{V} in \mathbb{G} . More precisely, each coset of f should correspond to some instruction encoded over a coset of t : $\forall j \in [n] \exists i \in [\ell]$ s.t. $f(\mu^j \mathbb{V}) = t(\omega^i \mathbb{V})$.

This type of relation can be abstracted as a *vector lookup*: we would like to prove that polynomial f faithfully “looks up” vectors encoded in the instruction table polynomial t . That is, only valid instructions are encoded. The standard approach for vector lookups build on lookup protocols for individual field elements [18, 35–37, 58, 70, 71]. An element lookup allows proving the simpler relation that every evaluation of a polynomial f_1 over \mathbb{G} exists in the evaluations of the table polynomial t_1 on \mathbb{H} . To build a vector lookup from an element lookup, one may encode each position $i \in [m]$ of the instruction vectors within a different polynomial, resulting in polynomials $[f_i]_{i \in [m]}$ and $[t_i]_{i \in [m]}$. If the polynomials are committed using a linearly-homomorphic commitment scheme, the verifier can sample a random challenge $\beta \leftarrow_s \mathbb{F}$ to randomly combine the position commitments. The prover and verifier jointly compute (commitments to) polynomials $\hat{f} = \sum_{i \in [m]} \beta^i \cdot f_i$ and $\hat{t} = \sum_{i \in [m]} \beta^i \cdot t_i$, and perform an element lookup with respect to \hat{f} and \hat{t} [37].

This approach to vector lookups incurs verification cost and proof size that is at least linear in the vector size m . Recall in the machine computation application, we are proposing encoding instruction constraints within a vector. At the very least, the simplest instructions encoding 64-bit arithmetic requires constraints on the order of 128; in richer instruction sets, such as EVM, instructions can be much larger (e.g., 1.5 million for ECDSA verification or 20000 for SHA256). Linear scaling in m of the vector lookup is at best a significant overhead and at worst is a prohibitive road block to richer instruction sets. Our first key technical contribution is the construction of new *succinct* vector lookup arguments for univariate and multivariate polynomials (Sect. 2.2) where proof size and verifier cost is succinct in n , ℓ , and vector size m .

The vector lookup proves the computation commitment f indeed includes encodings of valid instructions. Ideally, we would be able to directly apply an existing proof system to f to prove the validity of the executed computation. However, there are two additional hurdles to overcome (Sect. 2.3). First, f is constructed as a stitching together of the computation encodings for each of the individual n executed instructions. It is not necessarily the case (and in fact not the case for existing proof systems) that a direct stitching together of the “local” instruction computation encodings results in a valid computation encoding for the “global” sequence of instructions; it may be the case that some global structure is required in the computation commitment. Nevertheless, we provide a protocol for adapting f to f' to recover the global structure required in three existing polyIOPs, PLONK [38], Marlin [30], and HyperPLONK [27].

Lastly, applying a polyIOP directly to f' would not quite meet our succinctness goal. Recall, the verifier input to each instruction computation is $(inst_{in}, mem_{in}, inst_{out}, mem_{out})$. Thus, to verify the full executed computation, the verifier will need $[inst_j, mem_j]_j^n$ where the statement for the j^{th} instruction is $(inst_j, mem_j, inst_{j+1}, mem_{j+1})$. Instead, to enable succinctness, the verifier will hold only the input state to the first instruction $(inst_0, mem_0)$ and the output state of the last instruction $(inst_n, mem_n)$. We provide a protocol to prove the wellformedness of the intermediate instruction states, i.e., that the output state from instruction j is the same as the input state to instruction $j + 1$.

2.2 Contribution: Succinct Vector Lookup Arguments

The constructions we propose, `CosetLkup` for the univariate polynomial case and `SubcubeLkup` for the multivariate polynomial case, both derive from the following technical lemma of Haböck [45, Section 3.4] (reformulated in [22, Section 4.4]):

Lemma 1 (informal). *Suppose $\left[[f_{i,j}]_{j \in [m]} \right]_{i \in [n]}$ and $\left[[t_{i,j}]_{j \in [m]} \right]_{i \in [\ell]}$ are sequences of element vectors in field \mathbb{F} . Then, the vector lookup relation $\left\{ \{f_{i,j}\}_{j \in [m]} \right\}_{i \in [n]} \subseteq \left\{ [t_{i,j}]_{j \in [m]} \right\}_{i \in [\ell]}$ holds if and only if there exists a sequence of field elements $[c_i]_{i \in [\ell]}$ such that the following equality holds over the rational function field $\mathbb{F}(X, Y)$:*

$$\sum_{i \in [n]} 1/(X + \sum_{j \in [m]} f_{i,j} Y^j) = \sum_{i \in [\ell]} c_i / (X + \sum_{j \in [m]} t_{i,j} Y^j).$$

To achieve this equality, the field elements $[c_i]_{i \in [\ell]}$ are set to the counts that vector t_i appears in f . Intuitively, this lemma represents the logarithmic derivative of the polynomial equality $\prod_{i \in [n]} (X + \sum_{j \in [m]} f_{i,j} Y^j) = \prod_{i \in [\ell]} (X + \sum_{j \in [m]} t_{i,j} Y^j)^{c_i}$. Our protocols check this equality by evaluating the rational functions on random verifier challenges $\alpha, \beta \leftarrow_{\$} \mathbb{F}^2$. The challenge then is proving this equality succinctly to a verifier given the vector encodings.

$$\sum_{i \in [n]} 1/(\alpha + \sum_{j \in [m]} \beta^j \cdot f_{i,j}) = \sum_{i \in [\ell]} c_i / (\alpha + \sum_{j \in [m]} \beta^j \cdot t_{i,j}).$$

Consider again the univariate polynomial encoding of f with vectors encoded within cosets. We will build up to two polynomials U_f and U_t that encode the left and right sides of the equality expression above, respectively. Without loss of generality, consider the left side of the equality dealing with f where $f_{i,j} = f(\mu^i \gamma^j)$. That is, U_f will encode as its evaluations over \mathbb{G} :

$$\left[\left[U_f(\mu^i \gamma^j) = 1 / \left(\alpha + \sum_{k \in [m]} \beta^k \cdot f(\mu^i \gamma^k) \right) \right]_{i \in [n]} \right]_{j \in [m]}$$

Given U_f and an analogously-encoded U_t , the final equality is checked using a polynomial identity test known as a sum check, in which we compare the sum of U_f over \mathbb{G} and U_t over \mathbb{H} .

Now let us provide some details on how U_f is built up. The main challenge is proving that the summation in the denominator of U_f correctly encodes the elements of each vector. To do this, first consider the following helper polynomials I_f and S_f . Polynomial S_f directly encodes the claim summation for each vector within the coset for that vector. Polynomial I_f encodes the m powers-of- β in each coset.

$$\left[[I_f(\mu^i \gamma^j) = \beta^j]_{i \in [n]} \right]_{j \in [m]}, \quad \left[[S_f(\mu^i \gamma^j) = \sum_{k \in [m]} \beta^k \cdot f(\mu^i \gamma^k)]_{i \in [n]} \right]_{j \in [m]}$$

The encodings of I_f and S_f are proved to the verifier again using standard polynomial identities. In this case, m test protocols check that the following identities hold over some subgroup:

- $I_f(1) = 1$: The first element of I_f is anchored to equal to 1.
- $(I_f(\gamma X) - \beta \cdot I_f(X))(X - \gamma^{m-1}) = 0$ over \mathbb{V} : The term $(I_f(\gamma X) - \beta \cdot I_f(X))$ enforces the next element in the coset \mathbb{V} (generated by γ) is equal to β times the previous element. The last term $(X - \gamma^{m-1})$ excludes the last element which would carry-over to the first element: $\beta^{m-1} \cdot \beta \neq 1$. Since the first element was anchored to 1 and the last element is excluded, this sets the evaluations of \mathbb{V} to be equal to the powers of β .
- $(I_f(\mu X) - I_f(X)) \cdot Z_{\mu^{n-1}\mathbb{V}}(X) = 0$ over \mathbb{G} : The first term $(I_f(\mu X) - I_f(X))$ enforces the next element in \mathbb{G} is equal to the previous element in \mathbb{G} . The second term $Z_{\mu^{n-1}\mathbb{V}}(X)$ excludes the last coset where Z is the “vanishing polynomial” that evaluates to 0 on $\mu^{n-1}\mathbb{V}$. These checks ensure that the j^{th} element of every coset is the same. Since we know the powers of β are encoded in coset \mathbb{V} , this check enforces that the same powers are propagated to the other cosets in \mathbb{G} .

To prove the wellformedness of S_f , we introduce another helper polynomial B_f which encodes the partial summations of S_f and builds up to the claimed summation inductively:

$$\left[[B_f(\mu^i \gamma^j) = \sum_{k \in [j]} \left(\beta^k \cdot f(\mu^i \gamma^k) - \frac{S_f(\mu^i \gamma^j)}{m} \right)]_{i \in [n]} \right]_{j \in [m]}.$$

Then the polynomial identities to complete the wellformedness verification of S_f are as follows:

- $S_f(\gamma X) = S_f(X)$ over \mathbb{G} : Within a coset, the same claimed summation is encoded throughout, i.e., all of the evaluations over a coset are a constant (the respective summation).
- $B_f(\gamma X) = B_f(X) + I_f(\gamma X) \cdot f(\gamma X) - \frac{S_f(X)}{m}$ over \mathbb{G} : The inductive statement enforces that the next element in the partial summation sums the previous partial summation with the contribution of the next element in the coset, namely $\beta^k \cdot f(\mu^j \gamma^k)$ where the next power of β is encoded within I_f . Lastly, the normalized claimed summation, $\frac{S_f(X)}{m}$, is subtracted for every element in the coset. If the claimed sum is correct, then these subtractions will exactly cancel out with the true sum over the full coset.

Finally, given S_f , one last polynomial identity is used to prove the form of U_f : $U_f(X) \cdot (\alpha + S_f(X)) = 1$ over \mathbb{G} . Our use of coset encodings and of the subgroup generator γ to traverse cosets was critical in creating polynomial identities that can efficiently check this structure. Section 4 (Fig. 3) presents the full `CosetLkup` protocol. All together, when instantiated with the Marlin-KZG polynomial commitment scheme [30], `CosetLkup` admits a quasilinear prover, a constant-size proof, and constant pairing (and logarithmic field operations) verifier (see Fig. 2).

Extending to the Multivariate Setting. As shown before, the choice of encoding of vectors within the polynomial is important in enabling a succinct argument. Before, in the univariate setting, we chose to encode vectors within cosets of a multiplicative subgroup. For the multivariate setting, we encode vectors within subcubes of the boolean hypercube. Consider $(\log(mn))$ -variate polynomial f encoding n vectors of length m over the $(\log(mn))$ -dimension boolean hypercube $\{0, 1\}^{\log(mn)}$. The trailing $\log n$ bits select the vector and the leading $\log m$ bits select the vector position. Thus, for $[[f_{i,j}]_{j \in [m]}]_{i \in [n]}$: $[[f(j, i) = f_{i,j}]_{j \in \{0,1\}^{\log m}}]_{i \in \{0,1\}^{\log n}}$. The table polynomial t is encoded analogously over boolean hypercube $\{0, 1\}^{\log(m\ell)}$. We present a succinct vector lookup protocol `SubcubeLkup` for this subcube vector encoding over multivariate polynomials. It follows the same blueprint described above building polynomials I_f, S_f, B_f, U_f (respectively, U_t , etc.) and completes by checking the equality of Lemma 1 via a sum check over U_f and U_t . Our key trick to achieving succinctness in the univariate setting was using the coset substructure and defining polynomial identities that traverse a coset using the subgroup generator γ .

Recently, Diamond and Posen proposed new techniques that enable traversing subcubes of the boolean hypercube in an analogous manner [34]. They define a shift operator $\widetilde{\text{shft}}_b(f)$ that takes a μ -variate polynomial f , is parameterized by a subcube size b , and outputs a polynomial shifted by the subcube. That is, for all $i \in \{0, 1\}^{\mu-b}$ and for all $j \in \{0, 1\}^b$, it holds that $\widetilde{\text{shft}}_b(f)(j, i) = f(\text{bin}_b(\text{int}_b(j) + 1 \bmod 2^b), i)$ where int_b and bin_b map back and forth integers $[2^b]$ and boolean vectors $\{0, 1\}^b$. Using this operator, we can again define succinctly verifiable polynomial identities to check the wellformedness of U_f and U_t .

To illustrate this, consider the powers-of- β polynomial I_f . Now $I_f(X_{\lfloor \log(mn) \rfloor})$ is a $(\log(mn))$ -variate polynomial encoding the m powers of β in the boolean subcubes of its leading $\log m$ bits: $\left[[I_f(j, i) = \beta^{\text{int}_{\log m}(j)}]_{j \in \{0,1\}^{\log m}} \right]_{i \in \{0,1\}^{\log n}}$. The following polynomial identities checked via multivariate zero tests [27, 59] over $\{0, 1\}^{\log(mn)}$ verify the wellformedness of I_f :

- $(I_f(X_{\lfloor \log(mn) \rfloor}) - 1)(\tilde{e}q_{\log m}(X_{\lfloor \log m \rfloor}, 0_{\lfloor \log m \rfloor})) = 0$: The first term checks that I_f is anchored to 1. The second term $\tilde{e}q_{\log m}(X_{\lfloor \log m \rfloor}, 0_{\lfloor \log m \rfloor})$ enforces this check only for the first position ($0_{\lfloor \log m \rfloor}$) of each subcube. The polynomial $\tilde{e}q_{\log m}$ takes in two boolean vectors of length $\log m$ and outputs 1 if they are equal and 0 otherwise.
- $(\widetilde{\text{shft}}_{\log m}(\tilde{I}_b)(X_{\lfloor \log(md_b) \rfloor}) - \beta \cdot \tilde{I}_b(X_{\lfloor \log(md_b) \rfloor})) \cdot (1 - \tilde{e}q_{\log m}(X_{\lfloor \log m \rfloor}, 1_{\lfloor \log m \rfloor})) = 0$: The first term enforces the next element in the subcube (generated by $\widetilde{\text{shft}}$) is equal to β times the previous element. The last term excludes the last element of the subcube ($1_{\lfloor \log m \rfloor}$) preventing carry-over: $\beta^{m-1} \cdot \beta \neq 1$. As before, since the first element was anchored to 1 and the last element is excluded, this sets the evaluations of every subcube to be equal to the powers of β .

The full details of `SubcubeLkup` are given in the full version [33]. When instantiated with an appropriate polynomial commitment scheme, it enables an efficient linear-time prover with sublinear proof size and verifier (see Fig. 2).

2.3 Contribution: Vector Lookups for Machine Execution

We described our high level strategy of dynamically creating a computation commitment to only executed instructions through a vector lookup argument on the machine commitment table of valid instructions. There are two further challenges to overcome to apply existing polyIOP-based proof systems to this computation commitment: (1) the computation commitment may require some global structure that is lost by stitching together computation commitments for individual instructions, and (2) the NP statement for this computation commitment is not necessarily succinct.

Recovering Global Structure of the Computation Commitment. To illustrate the issue of global structure, let us examine the structure of a computation commitment for a specific polyIOP, PLONK [38]. PLONK is a polyIOP for the “Plonkish” arithmetization which is a natural encoding of computation in NP with a circuit-like structure [61]. Consider a simplified example of Plonkish for an arithmetic circuit with m gates. The computation trace is encoded as a vector of wire values $z \in \mathbb{F}^{3m}$:

$$z = \left[(z_0^{(l)}, z_0^{(r)}, z_0^{(o)}), (z_1^{(l)}, z_1^{(r)}, z_1^{(o)}), \dots, (z_{m-1}^{(l)}, z_{m-1}^{(r)}, z_{m-1}^{(o)}) \right].$$

We denote the ordering of vector z such that $(z_{3i}, z_{3i+1}, z_{3i+2}) = (z_i^{(l)}, z_i^{(r)}, z_i^{(o)})$ correspond to the left, right, and output wires of gate $i \in [m]$, respectively. The

computation is encoded by two vectors, $sel \in \mathbb{F}^m$ and $\sigma \in \mathbb{F}^{3m}$. The selector vector sel specifies the gate type by encoding 1 at index i if gate i is an addition gate and 0 for a multiplication gate. The copy vector σ specifies the connections of wires between gates by encoding a permutation of the indices $[3m]$ (i.e. $\{\sigma_i\}_{i \in [3m]} = \{i \in [3m]\}$). The copy vector is constructed such that wires that are connected have permuted indices. A computation trace satisfies a computation encoding if:

- *Gate constraints:* $\forall i \in [m], sel_i \cdot \left(z_i^{(l)} + z_i^{(r)} \right) + (1 - sel_i) \cdot z_i^{(l)} \cdot z_i^{(r)} = z_i^{(o)}$.
- *Copy constraints:* $\forall i \in [3m], z_i = z_{\sigma_i}$.

In the univariate polyIOP PLONK [38], $\bar{sel} \in \mathbb{F}^m[X]$ and $\bar{\sigma} \in \mathbb{F}^{3m}[X]$ are interpolated as polynomials fixing their evaluations over subgroups of appropriate size. Commitments to these polynomials form the computation commitment.

In the machine computation setting, each instruction $i \in [\ell]$ in the instruction set is represented by a pair of vectors $[sel_i \in \mathbb{F}^m, \sigma_i \in \mathbb{F}^{3m}]_{i \in [\ell]}$. Following our vector lookup strategy, these vectors are encoded in table polynomials $t sel$ and $t \sigma$ where each vector is encoded in cosets of \mathbb{V} within $\mathbb{H} = \langle \omega \rangle$. Assume, for simplicity, that $t sel$ and $t \sigma$ use the same evaluation subgroups \mathbb{V} and \mathbb{H} (where $|\mathbb{V}| = 3m$ and $|\mathbb{H}| = 3m\ell$), e.g., that $t sel$ repeats each gate selector three times to pad out. For a machine computation that executes n instructions $[inst_{i \in [\ell]}]_{i \in [n]}$, we define two new polynomials \bar{sel} and $\bar{\sigma}$ that encode the executed instruction vectors in the cosets \mathbb{V} within $\mathbb{G} = \langle \mu \rangle$ (where $|\mathbb{G}| = 3mn$). The form of these polynomials is proved exactly using a vector lookup as demonstrated before:

$$[\bar{sel}(\mu^i \mathbb{V}) = t sel(\omega^{inst_i} \mathbb{V})]_{i \in [n]}, \quad [\bar{\sigma}(\mu^i \mathbb{V}) = t \sigma(\omega^{inst_i} \mathbb{V})]_{i \in [n]}.$$

If we examine the resulting selector polynomial \bar{sel} defined over \mathbb{G} , we find that it fits the form needed for the PLONK polyIOP. The m gate selections for each of the n executed instructions are all encoded within \bar{sel} .

However, now consider the resulting copy polynomial $\bar{\sigma}$ defined over \mathbb{G} . Recall that the PLONK polyIOP expects a permutation over $|\mathbb{G}| = 3mn$ encoded within $\bar{\sigma}$. This is not the case for $\bar{\sigma}$; the global permutation structure is damaged. It is true that each instruction vector encodes a permutation over $[3m]$, and thus by the vector lookup, the evaluations of each coset of \mathbb{V} in $\bar{\sigma}$ encode a permutation over $[3m]$. Fortunately, we can recover the global permutation over $[3mn]$ by offsetting the permutation encoded in each coset $i \in [n]$ over $[3m]$ by $3mi$. Define the offset copy polynomial $\bar{\sigma}'$ and the offset polynomial s over \mathbb{G} :

$$[\bar{\sigma}'(\mu^i \mathbb{V}) = 3mi + \bar{\sigma}(\mu^i \mathbb{V})]_{i \in [n]}, \quad [s(\mu^i \mathbb{V}) = 3mi]_{i \in [n]}.$$

The prover uses the following polynomial identities to succinctly prove that the offset was performed correctly:

- $s(X) = 0$ over \mathbb{V} : The first coset is anchored to evaluate to 0; there is no offset needed.

- $(3m + s(X) - s(\mu X))(Z_{\mu^{n-1}\mathbb{V}}(X)) = 0$ over \mathbb{G} : The first term enforces the next element in \mathbb{G} is equal to the previous element offset by an additional $3m$. The second term excludes enforcing an additional offset from the last coset to the first coset. Since we know zero is encoded in coset \mathbb{V} , these checks enforce that each coset in sequence offsets by an additional $3m$.
- $\bar{\sigma}'(X) = \bar{\sigma}(X) + s(X)$ over \mathbb{G} : The offset copy polynomial adds the correct encoded offsets.

With the recovered global structure, the prover has now on-the-fly generated \bar{sel} and $\bar{\sigma}'$ which encode the executed machine computation and proved their correctness with respect to the machine commitments $t sel$ and $t \sigma$. The polyIOP PLONK can be applied directly.

Compressing the NP Statement of the Computation Commitment.

Our last challenge is to compress the statement for the unrolled computation commitment to allow for succinct verification. Naively, the statement for the unrolled computation commitment consists of the statements for each executed instruction: $[(inst_{in,i}, mem_{in,i}, inst_{out,i}, mem_{out,i})]_{i \in [n]}$. Not only does this prevent succinct verification, but it also prevents zero-knowledge of intermediate program execution state. We address this by observing that the verifier does not need to have the intermediate program state; it is sufficient for the verifier to simply check that the intermediate program state is passed correctly between instructions. That is, that $(inst_{out,i}, mem_{out,i}) = (inst_{in,i+1}, mem_{in,i+1})$ for all $i \in [n]$. Then the verifier only need hold the starting state $(inst_{in,0}, mem_{in,0})$ and the ending state $(inst_{out,n}, mem_{out,n})$.

In PLONK, the statement and witness for the unrolled computation are encoded together in a polynomial z (referred to as the extended witness) defined over \mathbb{G} . In which, the statement and witness for each instruction are encoded within a corresponding coset of \mathbb{V} in \mathbb{G} : for instruction $i \in [n]$, the evaluation of $z(\mu^i \mathbb{V}) = (inst_{in,i}, mem_{in,i}, inst_{out,i}, mem_{out,i}, w_i)$ for a corresponding witness vector w_i . More precisely, we define a subgroup $\mathbb{V}_x = \langle \psi \rangle \subset \mathbb{V}$ where $|\mathbb{V}_x| = 2 \cdot (|inst| + |mem|)$ whose cosets will encode the statement for each executed instruction. Further, we define subgroup $\mathbb{V}_{in} \subset \mathbb{V}_x$ and it's single coset \mathbb{V}_{out} such that $\mathbb{V}_{in} \cup \mathbb{V}_{out} = \mathbb{V}_x$ and $|\mathbb{V}_{in}| = |\mathbb{V}_{out}| = |\mathbb{V}_x|/2$. These will encode the input and output states of each executed instruction: $\forall i \in [n]$,

$$[z(\mu^i \mathbb{V}_{in}) = [inst_{in,i}, mem_{in,i}]]_{i \in [n]} \quad [z(\mu^i \mathbb{V}_{out}) = [inst_{out,i}, mem_{out,i}]]_{i \in [n]}$$

Thus, proving consistency of intermediate program states, again, reduces to proving polynomial identities over certain cosets. Namely, for $i \in [1, n]$, $z(\mu^i \mathbb{V}_{in}) = z(\mu^{i-1} \mathbb{V}_{out})$. The above coset equality constraints can be written as the polynomial identity: $z(X) = z(\mu^{-1} \psi X)$ over $\mathbb{G}_{in} \setminus \mathbb{V}_{in}$ where $\mathbb{G}_{in} = \bigcup_{j=0}^{n-1} \mu^j \mathbb{V}_{in}$. The full details of proving input-output correspondence and our full adaption of PLONK to machine computation, Mux- PLONK, is given in Sect. 5.

Extending to Other polyIOPs. The general recipe that we described using vector lookups to build Mux- PLONK is quite modular. We demonstrate the

generality by extending two other polyIOPs for use with machine computation, each offering various tradeoffs.

We build Mux- HyperPLONK building off of the multivariate HyperPLONK polyIOP [27]. As motivated earlier, multivariate polyIOPs enable linear-time provers and use of the efficient sum check protocol [54] for proving polynomial identities. In Mux- HyperPLONK, we employ our new multivariate succinct vector lookup SubcubeLkup and show how to translate the PLONK global permutation recovery and input-output consistency checks to the multivariate setting.

Next, we build Mux- Marlin building off the univariate Marlin polyIOP [30]. Whereas, Mux- PLONK and Mux- HyperPLONK encode instructions using the Plonkish arithmetization, Marlin uses a rank-1 constraint system (R1CS) arithmetization which offers encoding tradeoffs [61]. The full details are given in the full version [33].

3 Preliminaries

Sets and Vectors. For a positive numbers m and n with $m < n$, let $[m, n]$ denote the vector $[m, \dots, n - 1]$ and $[n]$ be shorthand for $[0, n]$. We use $[\cdot]$ and (\cdot) to denote ordered vectors, $\{\cdot\}$ to denote sets, and $\{\{\cdot\}\}$ to denote a multiset. Any of these operators can be expanded via a subscript, i.e., $[a_i]_{i=1}^n = [a_1, \dots, a_n]$.

Fields, Groups, and Polynomials. Define \mathbb{F} to be a scalar field of large prime order p . We will use \mathbb{H} and \mathbb{V} (and various subscripts) to denote multiplicative subgroups of \mathbb{F}^* . We may denote a generator γ for a subgroup as $\mathbb{H} = \langle \gamma \rangle$. We will also denote subgroups to be subgroups of each other, say \mathbb{V} is a subgroup of \mathbb{H} , denoted $\mathbb{V} \leq \mathbb{H}$. We require that all multiplicative groups we use are FFT-friendly and have smooth sizes [10], i.e., are a power of two. That is, we want $2^L | p - 1$ for some large integer L , so each divisor of 2^L (every power of 2 less than 2^L) gives exactly one subgroup whose order is the divisor by Lagrange’s theorem. Many common curves support these properties including BN382 and BLS12-381 [6].

When \mathbb{V} is a subgroup of \mathbb{H} , we will make use of the cosets of \mathbb{V} in \mathbb{H} . A coset of \mathbb{V} is defined by a field element offset $a \in \mathbb{F}$ as $\{av : v \in \mathbb{V}\}$, which we may denote as $a\mathbb{V}$. For multiplicative subgroup $\mathbb{H} = \langle \omega \rangle, \mathbb{V}$ where $\mathbb{V} \leq \mathbb{H}$, $|\mathbb{H}| = nm$, and $|\mathbb{V}| = m$, then $[\omega^i \mathbb{V}]_{i \in [n]}$ forms the n distinct cosets of \mathbb{V} in \mathbb{H} .

Let $\mathbb{F}^{\leq d}[X_{[\mu]}]$ be the set of μ -variate polynomials in indeterminate $X_0, \dots, X_{\mu-1}$ with coefficients in \mathbb{F} with degree less than or equal to d . Similarly, let $\text{Func}^{\mathbb{F}}[X_{[\mu]}]$ be the set of μ -variate functions over \mathbb{F} . We use $X_{[\mu]}$ as shorthand for expanding $X_0, \dots, X_{\mu-1}$. For polynomials $f \in \mathbb{F}[X_{[\mu]}]$ and some evaluation domain D , we use $f(D)$ as shorthand for expanding the vector $[f(d)]_{d \in D}$.

Univariate Polynomials. For an arbitrary set S , let the vanishing polynomial for S be $Z_S(X) = \prod_{s \in S} (X - s)$ such that it evaluates to 0 for $s \in S$. A Lagrange polynomial $L_{x,S}$ is a polynomial of degree $|S| - 1$ that evaluates to zero on $S \setminus \{x\}$ and has $L_{x,S}(x) = 1$. For cosets of a multiplicative group \mathbb{V} in \mathbb{H} , both the vanishing polynomial $Z_{\mathbb{V}}$ and the Lagrange polynomial $L_{x,\mathbb{V}}$ for $x \in \mathbb{V}$ have efficiently computable forms. The vanishing polynomial takes the form

$Z_{\omega^i \mathbb{V}}(X) = X^m - \omega^{im}$. The Lagrange polynomial takes the form $L_{x, \mathbb{V}}(X) = \frac{c_x Z_{\mathbb{V}}(X)}{X - x}$ where c_x is the Lagrange constant for x defined to be $\frac{1}{\prod_{y \in \mathbb{V}, x \neq y} x - y}$. Note that $c_x, \forall x \in \mathbb{V}$ can be precomputed in $O(|\mathbb{V}|)$ time. In particular, for $\mathbb{V} = \langle \beta \rangle$ and $x = \beta^i$, then $c_x = (\beta^i / |\mathbb{V}|)$.

Further preliminaries including formal definitions of zero-knowledge proofs and polynomial interactive oracle proofs (polyIOPs) along with constructions for building-block polyIOPs are deferred to the full version [33].

4 Succinct Vector Lookup

We propose two constructions for succinct vector lookup. One in which the vectors are encoded within univariate polynomials and the second in which the vectors are encoded within multivariate polynomials. Both constructions follow the same high level blueprint and their security is derived from the following main technical lemma from Haböck [45]:

Lemma 2. *Let \mathbb{F} be a field with $\text{char}(\mathbb{F}) > \max(d_0, d_1)$. Suppose $\{\{f_{i,j}\}_{j \in [m]}\}_{i \in [d_0]}$ and $\{\{t_{i,j}\}_{j \in [m]}\}_{i \in [d_1]}$ are sequences of element vectors in \mathbb{F} . Then, $\{\{f_{i,j}\}_{j \in [m]}\}_{i \in [d_0]} \subseteq \{\{t_{i,j}\}_{j \in [m]}\}_{i \in [d_1]}$ if and only if there exists a sequence of field elements $[c_i]_{i \in [d_1]}$ such that*

$$\sum_{i \in [d_0]} 1 / (X - \sum_{j \in [m]} f_{i,j} Y^j) = \sum_{i \in [d_1]} c_i / (X - \sum_{j \in [m]} t_{i,j} Y^j).$$

where equality holds over the rational function field $\mathbb{F}(X, Y)$.

In this section, we present the univariate polynomial vector encoding construction, **CosetLkup**, in which vectors are encoded on coset evaluation domains. The multivariate construction, **SubcubeLkup**, encodes vectors on boolean subcube evaluation domains; it is deferred to the full version [33].

Recall the univariate polynomial encoding for vectors from Sect. 2.2. Given a table of vectors $[[t_{i,j}]_{j \in [m]}]_{i \in [d_1]}$ and a list of claimed looked up vectors $[[f_{i,j}]_{j \in [m]}]_{i \in [d_0]}$, we consider polynomial encodings $t \in \mathbb{F}^{md_1}[X]$ and $f \in \mathbb{F}^{md_0}[X]$ as follows. Consider two subgroups $\mathbb{H}_0 = \langle \omega_0 \rangle \leq \mathbb{F}$ and $\mathbb{H}_1 = \langle \omega_1 \rangle \leq \mathbb{F}$ such that $|\mathbb{H}_0| = md_0$ and $|\mathbb{H}_1| = md_1$. Further consider the shared subgroup $\mathbb{V} \leq \mathbb{H}_0$ (and $\mathbb{V} \leq \mathbb{H}_1$) such that $|\mathbb{V}| = m$ and $\mathbb{V} = \langle \gamma = \omega_0^{d_0} = \omega_1^{d_1} \rangle$. The vectors are encoded as the evaluations of each coset of \mathbb{V} in \mathbb{H}_0 and \mathbb{H}_1 respectively:

$$\left[f(\omega_0^i \mathbb{V}) = [f_{i,j}]_{j \in [m]} \right]_{i \in [d_0]}, \quad \left[t(\omega_1^i \mathbb{V}) = [t_{i,j}]_{j \in [m]} \right]_{i \in [d_1]}.$$

Given this encoding, we consider the following vector lookup relation for univariate polynomials:

$$R_{\text{Vlkup}} = \left\{ \perp, ([f], [t]), (f, t) : \{f(\omega_0^i \mathbb{V})\}_{i \in [d_0]} \subseteq \{t(\omega_1^i \mathbb{V})\}_{i \in [d_1]} \right\}.$$

We walk through the main points of our construction in Sect. 2.2. The full construction is provided in Fig. 3.

$R_{\text{vkup}} = \left\{ \left(\perp, \left(\mathbb{H}_0 = \langle \omega_0 \rangle, \mathbb{H}_1 = \langle \omega_1 \rangle, \mathbb{V} = \langle \gamma = \omega_0^{d_0} = \omega_1^{d_1} \rangle, \llbracket f \rrbracket, \llbracket t \rrbracket \right), (f, t) \right) \right\}$ $: \{f(\omega_0^i \mathbb{V})\}_{i \in [d_0]} \subseteq \{t(\omega_1^i \mathbb{V})\}_{i \in [d_1]}$
<p>CosetLkup.P($\perp, (\mathbb{H}_0, \mathbb{H}_1, \mathbb{V}, \llbracket f \rrbracket, \llbracket t \rrbracket), (f, t)$) \leftrightarrow CosetLkup.V($\perp, (\mathbb{H}_0, \mathbb{H}_1, \mathbb{V}, \llbracket f \rrbracket, \llbracket t \rrbracket)$)</p> <p>(1) P computes, sends, and proves the wellformedness of the count polynomial c that encodes the counts $[c_i]_{i \in [d_1]}$ where c_i is the number of times the vector $t(\omega_1^i \mathbb{V})$ appears in f.</p> <p>(a) P sends c defined over \mathbb{H}_1 setting the evaluation to be constant in each coset: $\{\{c(\omega_1^i \gamma^j) = c_i\}_{i \in [d_1]}\}_{j \in [m]}$.</p> <p>(b) P and V engage in ZeroTest($\mathbb{H}_1, \mathbb{H}_1$) to prove every coset of \mathbb{V} in \mathbb{H}_1 encodes a constant value: $c(\gamma X) = c(X)$ over \mathbb{H}_1.</p> <p>(2) V sends random challenges $(\alpha, \beta) \in (\mathbb{F} \setminus (\mathbb{H}_0 \cup \mathbb{H}_1))^2$.</p> <p>(3) P computes and sends the position-indexing powers-of-β polynomial $I_b(X)$ for $b \in \{0, 1\}$ and proves its wellformedness:</p> <p>(a) P computes and sends I_b defined over \mathbb{H}_b setting the evaluation of the j^{th} element of each coset to be j-th power-of-β, β^j: $\{\{I_b(\omega_b^i \gamma^j) = \beta^j\}_{i \in [d_b]}\}_{j \in [m]}$</p> <p>(b) P and V engage in ZeroTest(\mathbb{V}, \mathbb{H}_b) to prove $L_{1, \mathbb{V}}(X)(I_b(X) - 1) = 0$ over \mathbb{V}.</p> <p>(c) P and V engage in ZeroTest(\mathbb{V}, \mathbb{H}_b) to prove $(I_b(\gamma X) - \beta \cdot I_b(X))(X - \gamma^{m-1}) = 0$ over \mathbb{V}.</p> <p>(d) P and V engage in ZeroTest($\mathbb{H}_b, \mathbb{H}_b$) to prove $(I_b(X) - I_b(\omega_b X))Z_{\omega_b^{d_b-1} \mathbb{V}}(X) = 0$ over \mathbb{H}_b.</p> <p>(4) P computes and sends the summation polynomial $S_b(X)$ for $b \in \{0, 1\}$ and proves its wellformedness:</p> <p>(a) P computes and sends S_b defined over \mathbb{H}_b setting the evaluation to be constant in each coset $\omega_b^i \mathbb{V}$ for $i \in [d_b]$ to summation of the values in the coset multiplied by powers of β. Let $p_0 = f$ and $p_1 = t$:</p> $\{\{S_b(\omega_b^i \gamma^j) = \sum_{k \in [m]} \beta^k \cdot p_b(\omega_b^i \gamma^k)\}_{i \in [d_b]}\}_{j \in [m]}$ <p>(b) P and V engage in ZeroTest($\mathbb{H}_b, \mathbb{H}_b$) to prove every coset of \mathbb{V} in \mathbb{H}_b encodes a constant: $S_b(\gamma X) = S_b(X)$ over \mathbb{H}_b.</p> <p>(5) P sends the induction polynomial $B_b(X)$ for $b \in \{0, 1\}$ and proves well-formedness. Let $p_0 = f$ and $p_1 = t$:</p> <p>(a) P computes and sends B_b defined over \mathbb{H}_b that accumulates the normalized summation:</p> $\{\{B_b(\omega_b^i \gamma^j) = \sum_{k \in [j]} (\beta^k \cdot p_b(\omega_b^i \gamma^k) - (S_b(\omega_b^i \gamma^j))/m)\}_{i \in [d_b]}\}_{j \in [m]}$ <p>(b) P and V engage in ZeroTest($\mathbb{H}_b, \mathbb{H}_b$) to prove induction $B_b(\gamma X) = (B_b(X) + I_b(\gamma X) \cdot p_b(\gamma X)) - S_b(X)/m$ over \mathbb{H}_b.</p> <p>(6) P computes and sends the inverse polynomial $U_b(X)$ for $b \in \{0, 1\}$ and proves its wellformedness:</p> <p>(a) P computes and sends U_b defined over \mathbb{H}_b setting the evaluation of each coset $\omega_b^i \mathbb{V}$ for $i \in [d_b]$ to the inverse of the summation and random challenge α as appears in the denominator of the Haböck lemma. Let $p_0 = f$ and $p_1 = t$:</p> $\{\{U_b(\omega_b^i \gamma^j) = 1/(\alpha - \sum_{k \in [m]} \beta^k p_b(\omega_b^i \gamma^k))\}_{i \in [d_b]}\}_{j \in [m]}$ <p>(b) P and V engage in ZeroTest($\mathbb{H}_b, \mathbb{H}_b$) to prove inversion: $U_b(X) \cdot (\alpha - S_b(X)) = 1$ over \mathbb{H}_b.</p> <p>(7) P proves summations of U_0 and $c \cdot U_1$ over \mathbb{H}_0 and \mathbb{H}_1, respectively, are equal. Let $u_0 = U_0$ and $u_1 = c \cdot U_1$. For $b \in \{0, 1\}$:</p> <p>(a) P interpolates and sends polynomials T_b over \mathbb{H}_b such that:</p> $\{T_b(\omega_b^i) = \sum_{k=i}^{ \mathbb{H}_b -1} u_0(\omega_b^k)\}_{i \in [\mathbb{H}_b]}$ <p>(b) P and V engage in ZeroTest($\mathbb{H}_b, \mathbb{H}_b$) to prove $(X - \omega_b^{ \mathbb{H}_b -1})(T_b(\omega_b X) + u_b(X) - T_b(X)) = 0$ over \mathbb{H}_b.</p> <p>(c) P and V engage in ZeroTest($\mathbb{H}_b, \mathbb{H}_b$) to prove $L_{\omega_b^{ \mathbb{H}_b -1}, \mathbb{H}_b}(X)(T_b(X) - u_b(X)) = 0$ over \mathbb{H}_b.</p> <p>(d) P and V engage in ZeroTest($\mathbb{H}_0, \mathbb{H}_0 \cup \mathbb{H}_1$) to prove $L_{1, \mathbb{H}_0}(X)(T_0(X) - T_1(X)) = 0$ over \mathbb{H}_0.</p>

Fig. 3. Vector lookup argument in which vectors are encoded as evaluations over coset domains in a univariate polynomial.

Security. We prove the completeness, knowledge soundness, and zero knowledge of `CosetLkup` in the following two theorems. The zero-knowledge of `CosetLkup` is achieved through our zero-knowledge compiler observing that `CosetLkup` is domain-restriction admissible and reduces to zero test polynomial identities.

Theorem 1. *CosetLkup for R_{vlkup} (Fig. 3) is complete and knowledge sound with negligible error.*

We defer the proof along with the concrete accounting of adversary advantage to the full version [33].

Remark 1 (Perfect Completeness). `CosetLkup` fails to achieve perfect completeness since the chosen randomness may make the denominator $(\alpha - \sum_{j \in [m]} f_{i,j} \beta^j) = 0$ or $(\alpha - \sum_{j \in [m]} t_{i,j} \beta^j) = 0$, so the fraction is undefined. Protostar [22] illustrates a technique to recover perfect completeness in exchange for slightly increased soundness error. When the evaluations of U_b are undefined at $\omega_b^i \gamma^j$, the prover sets $U_b(\omega_b^i \gamma^j) = 0$. At step 6(b), instead of checking $U_b(X) \cdot (\alpha - S_b(X)) \stackrel{?}{=} 1$ over \mathbb{H}_b , verifier instead checks $(U_b(X) \cdot (\alpha - S_b(X)) - 1)(\alpha - S_b(X)) \stackrel{?}{=} 0$ over \mathbb{H}_b . Then either $U_b(\omega_b^i \gamma^j) = \frac{1}{\alpha - S_b(\omega_b^i \gamma^j)} = \frac{1}{\alpha - \sum_{k \in [m]} p_b(\omega_b^i \gamma^k) \beta^k}$ or $\alpha - S_b(\omega_b^i \gamma^j) = 0$. The latter case captures the undefined scenario. The soundness error is raised by the two additional zero tests.

Theorem 2. *The compiled polyIOP using the compiler in the full version [33] of CosetLkup for R_{vlkup} (Fig. 3) is honest-verifier zero-knowledge.*

We defer the proof to the full version [33].

k -Vector Lookup. We can extend the lookup argument to work across k pairs of polynomials (f_i, t_i) for $i \in [k]$ checking that the same vector lookup applies across all k pairs. Our approach follows the multitable approach in [37] simply using a random linear combination to construct an expanded hash combining evaluations from all k polynomials. To be precise, prover sends f as the random linear combination of $(f_i)_{i \in [k]}$ and t as the linear combination of $(t_i)_{i \in [k]}$ using verifier randomness. Then the prover and verifier engage in `CosetLkup` using f and t . The relation is captured as:

$$R_{k\text{-vlkup}} = \left\{ (\perp, \llbracket f \rrbracket, \llbracket t \rrbracket\rrbracket_{i \in [k]}, (f_i, t_i)_{i \in [k]} : \{(f_i(\omega_0^j \mathbb{V}))_{i \in [k]}\}_{j \in [d_0]} \subseteq \{(t_i(\omega_1^j \mathbb{V}))_{i \in [k]}\}_{j \in [d_1]}\} \right\}.$$

Corollary 1. *k - CosetLkup for $R_{k\text{-vlkup}}$ is complete and knowledge sound with negligible error, and the compiled polyIOP using the compiler in the full version [33] is honest-verifier zero-knowledge.*

5 Succinct Arguments for Unrolled Machine Execution from Vector Lookups

We model a machine execution of a machine with ℓ instructions using ℓ indices $[i_i]_{i=0}^{\ell-1}$ to an indexed relation R (e.g., rank-1 constraint satisfiability or circuit satisfiability). The index for an instruction takes in a statement x of the form:

$$x = (inst_{in}, mem_{in}, inst_{out}, mem_{out}),$$

which can be parsed as two parts. The first part $(inst_{in}, mem_{in})$ is the “input” to the instruction where $inst_{in} \in \mathbb{Z}_\ell$ specifies which instruction to run and mem_{in} captures the current memory (or state) of the machine. The second part $(inst_{out}, mem_{out})$ is the “output” of the instruction specifying the next instruction to run ($inst_{out}$) and the resulting memory from executing the instruction (mem_{out}). We require that the indexed relation R enforces $inst_{in}$ to match the instruction index, i.e., that

$$\forall i \in \mathbb{Z}_\ell \ (i, (inst_{in}, mem_{in}, inst_{out}, mem_{out}), w) \in R \Rightarrow inst_{in} = i.$$

In this way, our formal modeling of machine execution ties together the control logic of determining the next instruction to run and the instruction logic of applying changes to memory. In the indexed relations that we consider (rank-1 constraint systems and circuit satisfiability), the index can easily be adjusted to enforce the above by including an equality check against a constant.

Given a set of instruction indices $[i_i]_{I=0}^{\ell-1}$ that satisfy the above, we define relation $R_{ME_{\text{exe}}, n}[R]$ for n steps of unrolled machine computation:

$$R_{ME_{\text{exe}}, n}[R] = \left\{ \left(\begin{array}{l} [i_i]_{I=0}^{\ell-1}, \\ (inst_0, mem_0, inst_n, mem_n), \\ ([inst_j, mem_j, w_j]_{j=0}^n) \end{array} \right) : \bigwedge_{j=0}^{n-1} (i_{inst_j}, (inst_j, mem_j, inst_{j+1}, mem_{j+1}), w_j) \in R \right\}$$

In the following sections we build unrolled machine execution proof systems for instructions encoded as rank-1 constraint systems ($R_{ME_{\text{exe}}, n}[R_{r1cs}]$) derived from the Marlin proof system [30].

Capturing Zero-Knowledge of Program Execution. Even with a zero-knowledge proof system for the above relation, membership in the relation can leak information about the number of execution steps, the starting and ending instructions, and possibly the program description if it is included in the memory state. An upper bound on the number of execution steps is a fundamental leakage of the unrolled execution proving approach. To mitigate leakage of starting and end instructions, we propose including special instructions for program start and successful return. Lastly, to mitigate leakage of program description, the memory state can be considered in two parts, one that includes the input and output registers that can be revealed to the verifier and another as a hiding commitment to the program description.

5.1 Mux-PLONK: Adapting the PLONK PolyIOP to Machine Execution

PLONK [38] is a polyIOP for NP statements encoded using the following PLONK arithmetization.

Definition 1. A PLONK relation is indexed by the tuple $(\mathbb{F}, sel, \sigma, G, \ell_s, \ell_z, d, d_x)$ where $sel = [[sel_{i,j}]_{i \in [d]}]_{j \in [\ell_s]}$ is the selector vector, the copy vector $\sigma : [d\ell_z] \rightarrow [d\ell_z]$ is a permutation over $d\ell_z$, and $G : \mathbb{F}[X_{[\ell_s, \ell_z]}]$ is the gate polynomial. The statement $x \in \mathbb{F}^{d_x}$ and witness $w \in \mathbb{F}^{d\ell_z - d_x}$ together form an input vector $z = [[z_{i,j}]_{i \in [d]}]_{j \in [\ell_z]}$ where the following algebraic relation encoding the gate constraints and copy constraints is satisfied:

$$R_{\text{plonk}} = \left\{ \left(\begin{array}{l} (\mathbb{F}, sel, \sigma, G, \ell_s, \ell_z, d, d_x), \\ x, \\ w \end{array} \right) : \begin{array}{l} z = x \parallel w \in \mathbb{F}^{d \times \ell_z} \\ \bigwedge_{i \in [d]} G([sel_{i,j}]_{j \in [\ell_s]}, [z_{i,j}]_{j \in [\ell_z]}) = 0 \\ \bigwedge_{i \in [d\ell_z]} z_{[i/d], imodd} = z_{[\sigma(i)/d], \sigma(i)modd} \end{array} \right\}$$

Here, we modify the PLONK relation to fit the univariate polyIOP setting.

$$\left\{ \left(\begin{array}{l} (\mathbb{F}, \mathbb{H}, \mathbb{H}_x, sel, \sigma, G, \ell_s, \ell_z, d, d_x), \\ ([x], \mathbb{K}), \\ (w, x) \end{array} \right) : \begin{array}{l} x(\mathbb{H}_x) = x \\ z = x \parallel w \in \mathbb{F}^{d \times \ell_z} \\ \bigwedge_{i \in [d]} G([sel_{i,j}]_{j \in [\ell_s]}, [z_{i,j}]_{j \in [\ell_z]}) = 0 \\ \bigwedge_{i \in [d\ell_z]} z_{[i/d], imodd} = z_{[\sigma(i)/d], \sigma(i)modd} \end{array} \right\}$$

Here, we take as part of indexing two groups \mathbb{H} and \mathbb{H}_x of size d and d_x , respectively. The statement is encoded as evaluations over domain \mathbb{H}_x for a polynomial x given to the verifier.

There are two sets of polynomials output as part of the computation commitment during indexing: (1) selector polynomials that encode the gate constraints, and (2) permutation polynomials that encode the copy constraints (i.e., which input value should be set equal to each other). As discussed in the technical overview (Sect. 2.3), in the application to machine execution, the values encoded in each of these polynomials for the computation commitment of a single instruction is instead encoded in a polynomial representing the entire instruction set, i.e., a machine commitment. Each instruction is encoded within a different coset of the machine commitment polynomial’s evaluation domain.

Define the following evaluation domains:

- Define $\mathbb{V} = \langle \gamma \rangle$ as the multiplicative subgroup of size d .
- Define $\mathbb{G} = \langle \mu \rangle$ as the multiplicative subgroup of size dn where n is the number of unrolled execution steps. Denote the n cosets of \mathbb{V} in \mathbb{G} as $[\mu^i \mathbb{V}]_{i=0}^{n-1}$.
- Define $\mathbb{H} = \langle \omega \rangle$ as the multiplicative subgroup of size $d\ell$ where ℓ is the number of instructions. Denote the ℓ cosets of \mathbb{V} in \mathbb{H} as $[\omega^i \mathbb{V}]_{i=0}^{\ell-1}$.
- Define $\mathbb{V}_x \leq \mathbb{V}$ as the multiplicative subgroup of size d_x where $d/d_x = a$.
- Define $\mathbb{V}_{in} \leq \mathbb{V}_x$ as the multiplicative subgroup of size $d_x/2$ generated by $\mu^{\frac{2nd}{d_x}}$ and $\mathbb{V}_{out} = \mu^{\frac{2nd}{d_x}} \mathbb{V}_{in}$ as the other coset of \mathbb{V}_{in} in \mathbb{V}_x encoding the two parts of the machine execution statement.

Using this notation, Fig. 4 provides details of the indexer for Mux- PLONK. Table selector polynomials $[tsel_j]_{j \in [\ell_s]}$ and table permutation polynomials $[t\sigma_j]_{j \in [\ell_z]}$ encode the full instruction set, encoding each instruction $i \in [\ell]$ within

$$\text{RMExe},n[\text{Rplonk}] = \left\{ \begin{array}{l} \left(\left(\left(\mathbb{F}, \mathbb{G}, \mathbb{H}, \mathbb{V}, \mathbb{V}_x, \mathbb{V}_{in}, G, \ell_s, \ell_z, [\text{sel}_i, \sigma_i]_{i \in [\ell]} \right), \right) \right. \\ \left. \left(\llbracket x_0 \rrbracket, \llbracket x_n \rrbracket \right), \right. \\ \left. \left(\text{inst}_j, \text{mem}_j, \mathbf{w}_j \right)_{j \in [n]}, x_0, x_n \right) \\ \\ x_0(\mathbb{V}_{in}) = [\text{inst}_0, \text{mem}_0] \\ x_n(\mathbb{V}_{out}) = [\text{inst}_n, \text{mem}_n] \\ \\ \vdots \\ \bigwedge_{j \in [n]} \left(\left(\mathbb{F}, \mathbb{V}, \mathbb{V}_x, \text{sel}_{\text{inst}_j}, \sigma_{\text{inst}_j}, G \right), \right. \\ \left. \left(\text{inst}_j, \text{mem}_j, \text{inst}_{j+1}, \text{mem}_{j+1} \right), \right. \\ \left. \mathbf{w}_j \right) \in \text{Rplonk} \end{array} \right\}$$

Mux-PLONK.Setup(λ): Return \perp

Mux-PLONK.Index($\perp, (\mathbb{F}, \mathbb{G}, \mathbb{H}, \mathbb{V}, \mathbb{V}_x, \mathbb{V}_{in}, G, \ell_s, \ell_z, [\text{sel}_i, \sigma_i]_{i \in [\ell]})$)

(1) Compute the prover parameter polynomials for each instruction index using the PLONK indexer.

$$\left[\begin{array}{l} \left(\begin{array}{l} pp_i \leftarrow ([\text{sel}_{i,j}]_{j \in [\ell_s]}, [\sigma_{i,j}]_{j \in [\ell_z]}, G), \\ vp_i \leftarrow ([[\text{sel}_{i,j}]]_{j \in [\ell_s]}, [[\sigma_{i,j}]]_{j \in [\ell_z]}, G) \end{array} \right) \\ \leftarrow \text{PLONK.Index}(\perp, (\mathbb{F}, \mathbb{V}, \mathbb{V}_x, \text{sel}_i, \sigma_i, G, \ell_s, \ell_z)) \end{array} \right]_{i \in [\ell]}$$

(2) Construct table selector polynomials $[\text{tsel}_j]_{j \in [\ell_s]}$ and table permutation polynomials $[\text{t}\sigma_j]_{j \in [\ell_z]}$ over \mathbb{H} by setting the evaluations of the cosets $[\omega^i \mathbb{V}]_{i \in [\ell]}$:

$$\left[[\text{tsel}_j(\omega^i \mathbb{V}) = \text{sel}_{i,j}(\mathbb{V})]_{i \in [\ell]} \right]_{j \in [\ell_s]} \quad \left[[\text{t}\sigma_j(\omega^i \mathbb{V}) = \sigma_{i,j}(\mathbb{V})]_{i \in [\ell]} \right]_{j \in [\ell_z]}$$

(3) Return $(pp \leftarrow ([\text{tsel}_j]_{j \in [\ell_s]}, [\text{t}\sigma_j]_{j \in [\ell_z]}, G), vp \leftarrow ([[\text{tsel}_j]]_{j \in [\ell_s]}, [[\text{t}\sigma_j]]_{j \in [\ell_z]}, G))$

Fig. 4. Mux- PLONK: Setup algorithm encoding PLONK computation commitment values for each instruction into cosets of an evaluation domain for the machine commitment.

coset $\omega^i \mathbb{V}$ of \mathbb{H} . Figure 5 then provides details of the proving protocol. A vector lookup (Sect. 4) is employed to lookup the appropriate executed instructions and encode them within new selector polynomials $[\text{sel}_j]_{j \in [\ell_s]}$ and new permutation polynomials $[\sigma'_j]_{j \in [\ell_z]}$ where now each executed instruction is encoded within coset $\mu^i \mathbb{V}$ of \mathbb{G} for $i \in [n]$. As discussed in the overview (Sect. 2.3), the resulting permutation polynomials $[\sigma'_j]_{j \in [\ell_z]}$ are malformed in that they no longer encode a permutation: all cosets evaluate to $[d\ell_z]$. Permutation polynomials $[\sigma_j]_{j \in [\ell_z]}$ are constructed to offset the evaluations of each coset $\mu^i \mathbb{V}$ by $i d\ell_z$ to recover the permutation.

Given these vector-lookup constructed (and edited) PLONK index polynomials, we are almost ready to apply PLONK directly to a polynomial x encoding

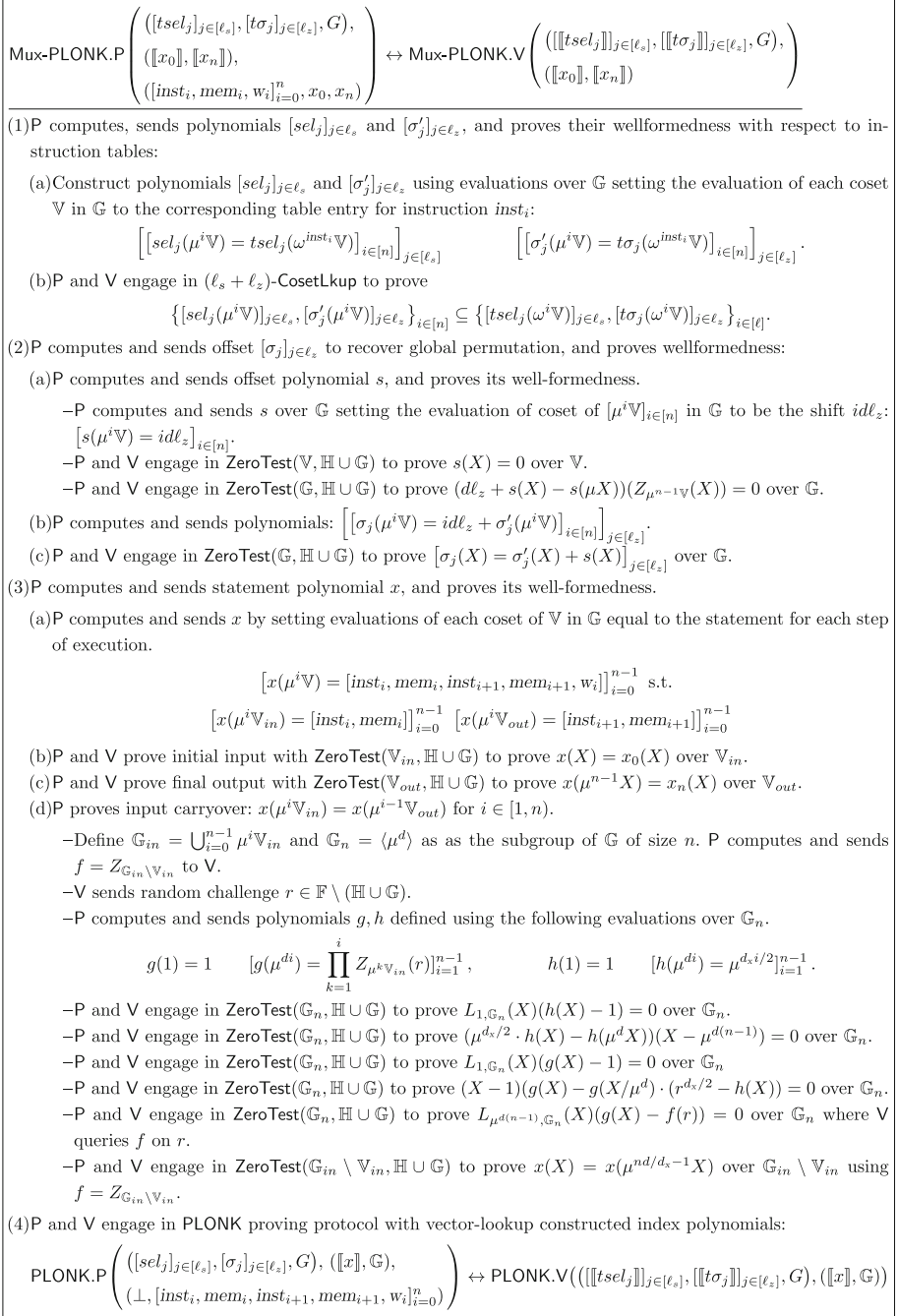


Fig. 5. Mux- PLONK: PLONK polyIOP with vector lookup for machine execution.

the witnesses of the executed machine computation. The last step is to prove that polynomial x correctly encodes the input-output correspondence of each sequence of instructions. We highlight a technical difficulty that arises in this step.

In proving the correspondence of inputs, we ask the prover to perform a **ZeroTest** over the set $\mathbb{G}_{in} \setminus \mathbb{V}_{in}$ where $\mathbb{G}_{in} = \bigcup_{i=0}^{n-1} \mu^i \mathbb{V}_{in}$. However, notice that $Z_{\mathbb{G}_{in} \setminus \mathbb{V}_{in}}(X) = \prod_{i \in [n]} Z_{\mu^i \mathbb{V}_{in}}(X)$ is not succinct and the fastest algorithm to compute this product incurs $O(|\mathbb{G}_{in} \setminus \mathbb{V}_{in}| \log^2(|\mathbb{G}_{in} \setminus \mathbb{V}_{in}|))$ cost [36] which cannot be afforded by the verifier. Instead, we ask the prover to send $f = Z_{\mathbb{G}_{in} \setminus \mathbb{V}_{in}}$ and prove that f satisfies the properties of a vanishing polynomial. One way is to evaluate f at some random point r and check if it agrees with $Z_{\mathbb{G}_{in} \setminus \mathbb{V}_{in}}(r)$. Recall that $[Z_{\mu^i \mathbb{V}_{in}}(X) = X^{d_x i/2} - \mu^{d_x i/2}]_{i \in [n]}$ since \mathbb{V}_{in} is of order $d_x/2$. To prove wellformedness, the prover creates new polynomials to accumulate the products of $Z_{\mu^i \mathbb{V}_{in}}(r)$ using induction. Since there are $n - 1$ terms in multiplication, we define group $\mathbb{G}_n = \langle \mu^d \rangle$ of order n to capture each intermediate result of the multiplication and skip the first element. To be concrete, the prover computes a polynomial h to encode the constant factor $\mu^{d_x i/2}$ of $Z_{\mu^i \mathbb{V}_{in}}(r)$ at μ^{di} , and a polynomial g to encode the intermediate product up to i^{th} item in multiplication $\prod_{k=1}^i Z_{\mu^k \mathbb{V}_{in}}(r)$ at μ^{di} . Then we use standard induction techniques to prove the induction is correct over \mathbb{G}_n skipping the first element:

- $L_{1, \mathbb{G}_n}(X)(h(X) - 1) = 0$ over \mathbb{G}_n : $h(1) = 1$ as the start of the induction.
- $(\mu^{d_x/2} \cdot h(X) - h(\mu^d X))(X - \mu^{d(n-1)}) = 0$ over \mathbb{G}_n : The next element in \mathbb{G}_n is equal to $\mu^{d_x/2}$ times the previous element excluding the last one. Since the first element is set to 1, this ensures that each element is set to the next power of $\mu^{\frac{d_x}{2}}$.
- $L_{1, \mathbb{G}_n}(X)(g(X) - 1) = 0$ over \mathbb{G}_n : $g(1) = 1$ as the starting of the induction.
- $(X - 1)(g(X) - g(X/\mu^d) \cdot (r^{d_x/2} - h(X))) = 0$ over \mathbb{G}_n : This enforces that the g 's evaluation on the i^{th} element (in \mathbb{G}_n) is equal to $r^{d_x/2} - h(X) = r^{d_x/2} - \mu^{d_x i/2}$ multiplied by g 's evaluation on the $(i - 1)^{th}$ element. The check excludes the last and the first one element. Since the first element is set to 1, this ensures that i^{th} element is set to the accumulated product $\prod_{k=1}^i Z_{\mu^k \mathbb{V}_{in}}(r)$ up to i .

Finally, the prover can prove $f(r) = g(\mu^{d(n-1)}) = \prod_{i \in [n]} Z_{\mu^i \mathbb{V}_{in}}(r)$ by evaluating $g(X)$ at $\mu^{d(n-1)}$ using Lagrange polynomial and query $f(r)$.

Security. We prove the completeness, knowledge soundness, and zero knowledge of Mux- PLONK in the following two theorems. The zero-knowledge of Mux- PLONK is achieved through our zero-knowledge compiler observing that Mux- PLONK is domain-restriction admissible with one exception that we handle explicitly.

Theorem 3. *Mux- PLONK for $R_{\text{MExe}, n}[R_{\text{plonk}}]$ (Fig. 5) is complete and knowledge sound with negligible error*

We defer the proof along with the concrete accounting of adversary advantage to the full version [33].

Theorem 4. *The compiled polyIOP using the compiler in the full version [33] of Mux- PLONK for $R_{ME_{x,n}}[R_{plonk}]$ (Fig. 5) is honest-verifier zero-knowledge.*

We defer the proof to the full version [33].

Acknowledgments. Zijing Di and Lucas Xia were funded by Stanford IOG Research Hub. Wilson Nguyen was partially funded by NSF, DARPA, the Simons Foundation, and NTT Research. Nirvan Tyagi was supported by NSF grant CNS-2120651 and by the Stanford Future of Digital Currency Initiative (FDCI). Any opinions, findings, and conclusions or recommendations expressed in the material are those of the authors and do not necessarily express reflect the views of DARPA and funding parties listed.

References

1. Polygon zkEVM, <https://wiki.polygon.technology/docs/zkEVM/introduction>
2. Risc zero, <https://www.risczero.com/docs/explainers>
3. zksync, <https://v2-docs.zksync.io/dev/>
4. zkWasm, <https://github.com/DelphinusLab/zkWasm>
5. Abe, M., Ohkubo, M., Suzuki, K.: 1-out-of- n signatures from a variety of keys. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 2501, pp. 415–432. Springer (2002)
6. Aranha, D.F., Housni, Y.E., Guillevic, A.: A survey of elliptic curves for proof systems. IACR Cryptol. ePrint Arch. p. 586 (2022), <https://eprint.iacr.org/2022/586>
7. Arun, A., Setty, S.T.V., Thaler, J.: Jolt: Snarks for virtual machines via lookups. IACR Cryptol. ePrint Arch. p. 1217 (2023)
8. Attema, T., Cramer, R., Fehr, S.: Compressing proofs of k -out-of- n partial knowledge. In: CRYPTO (4). Lecture Notes in Computer Science, vol. 12828, pp. 65–91. Springer (2021)
9. Baum, C., Malozemoff, A.J., Rosen, M.B., Scholl, P.: Mac’n’cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In: CRYPTO (4). Lecture Notes in Computer Science, vol. 12828, pp. 92–122. Springer (2021)
10. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: Snarks for C: verifying program executions succinctly and in zero knowledge. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 8043, pp. 90–108. Springer (2013)
11. Ben-Sasson, E., Chiesa, A., Goldberg, L., Gur, T., Riabzev, M., Spooner, N.: Linear-size constant-query iops for delegating computation. In: TCC (2). Lecture Notes in Computer Science, vol. 11892, pp. 494–521. Springer (2019)
12. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Scalable zero knowledge via cycles of elliptic curves. In: CRYPTO (2). Lecture Notes in Computer Science, vol. 8617, pp. 276–294. Springer (2014)
13. Ben-Sasson, E., Chiesa, A., Tromer, E., Virza, M.: Succinct non-interactive zero knowledge for a von neumann architecture. In: USENIX Security Symposium. pp. 781–796. USENIX Association (2014)
14. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: ITCS. pp. 326–349. ACM (2012)
15. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: Recursive composition and bootstrapping for SNARKS and proof-carrying data. In: STOC. pp. 111–120. ACM (2013)

16. Blum, M., Evans, W.S., Gemmell, P., Kannan, S., Naor, M.: Checking the correctness of memories. In: FOCS. pp. 90–99. IEEE Computer Society (1991)
17. Boneh, D., Drake, J., Fisch, B., Gabizon, A.: Halo infinite: Proof-carrying data from additive polynomial commitments. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 12825, pp. 649–680. Springer (2021)
18. Bootle, J., Cerulli, A., Groth, J., Jakobsen, S.K., Maller, M.: Arya: Nearly linear-time zero-knowledge proofs for correct program execution. In: ASIACRYPT (1). Lecture Notes in Computer Science, vol. 11272, pp. 595–626. Springer (2018)
19. Bowe, S., Chiesa, A., Green, M., Miers, I., Mishra, P., Wu, H.: ZEXE: enabling decentralized private computation. In: IEEE Symposium on Security and Privacy. pp. 947–964. IEEE (2020)
20. Bowe, S., Grigg, J., Hopwood, D.: Halo: Recursive proof composition without a trusted setup. IACR Cryptol. ePrint Arch. p. 1021 (2019)
21. Braun, B., Feldman, A.J., Ren, Z., Setty, S.T.V., Blumberg, A.J., Walfish, M.: Verifying computations with state. In: SOSP. pp. 341–357. ACM (2013)
22. Bünz, B., Chen, B.: Protostar: Generic efficient accumulation/folding for special sound protocols. IACR Cryptol. ePrint Arch. p. 620 (2023)
23. Bünz, B., Chiesa, A., Lin, W., Mishra, P., Spooner, N.: Proof-carrying data without succinct arguments. In: CRYPTO (1). Lecture Notes in Computer Science, vol. 12825, pp. 681–710. Springer (2021)
24. Bünz, B., Chiesa, A., Mishra, P., Spooner, N.: Recursive proof composition from accumulation schemes. In: TCC (2). Lecture Notes in Computer Science, vol. 12551, pp. 1–18. Springer (2020)
25. Bünz, B., Fisch, B., Szepieniec, A.: Transparent snarks from DARK compilers. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 12105, pp. 677–706. Springer (2020)
26. Campanelli, M., Faonio, A., Fiore, D., Li, T., Lipmaa, H.: Lookup arguments: Improvements, extensions and applications to zero-knowledge decision trees. IACR Cryptol. ePrint Arch. p. 1518 (2023)
27. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: Hyperplonk: Plonk with linear-time prover and high-degree custom gates. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 14005, pp. 499–530. Springer (2023)
28. Chen, M., Chiesa, A., Gur, T., O’Connor, J., Spooner, N.: Proof-carrying data from arithmetized random oracles. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 14005, pp. 379–404. Springer (2023)
29. Chen, M., Chiesa, A., Spooner, N.: On succinct non-interactive arguments in relativized worlds. In: EUROCRYPT (2). Lecture Notes in Computer Science, vol. 13276, pp. 336–366. Springer (2022)
30. Chiesa, A., Hu, Y., Maller, M., Mishra, P., Vesely, P., Ward, N.P.: Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In: EUROCRYPT (1). Lecture Notes in Computer Science, vol. 12105, pp. 738–768. Springer (2020)
31. Choudhuri, A.R., Garg, S., Goel, A., Sekar, S., Sinha, R.: Sublonk: Sublinear prover plonk. IACR Cryptol. ePrint Arch. p. 902 (2023)
32. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: CRYPTO. Lecture Notes in Computer Science, vol. 839, pp. 174–187. Springer (1994)
33. Di, Z., Xia, L., Nguyen, W.D., Tyagi, N.: Muxproofs: Succinct arguments for machine computation from vector lookups. IACR Cryptol. ePrint Arch. p. 974 (2023)
34. Diamond, B.E., Posen, J.: Succinct arguments over towers of binary fields. IACR Cryptol. ePrint Arch. p. 1784 (2023)

35. Eagen, L., Fiore, D., Gabizon, A.: cq: Cached quotients for fast lookups. *IACR Cryptol. ePrint Arch.* p. 1763 (2022)
36. Gabizon, A., Khovratovich, D.: flookup: Fractional decomposition-based lookups in quasi-linear time independent of table size. *IACR Cryptol. ePrint Arch.* p. 1447 (2022)
37. Gabizon, A., Williamson, Z.J.: plookup: A simplified polynomial protocol for lookup tables. *IACR Cryptol. ePrint Arch.* p. 315 (2020)
38. Gabizon, A., Williamson, Z.J., Ciobotaru, O.: PLONK: permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge. *IACR Cryptol. ePrint Arch.* p. 953 (2019)
39. Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: *STOC*. pp. 99–108. ACM (2011)
40. Goel, A., Green, M., Hall-Andersen, M., Kaptchuk, G.: Stacking sigmas: A framework to compose ζ -protocols for disjunctions. In: *EUROCRYPT (2)*. *Lecture Notes in Computer Science*, vol. 13276, pp. 458–487. Springer (2022)
41. Goel, A., Hall-Andersen, M., Kaptchuk, G.: Dora: Processor expressiveness is (nearly) free in zero-knowledge for RAM programs. *IACR Cryptol. ePrint Arch.* p. 1749 (2023)
42. Goel, A., Hall-Andersen, M., Kaptchuk, G., Spooner, N.: Speed-stacking: Fast sub-linear zero-knowledge proofs for disjunctions. *IACR Cryptol. ePrint Arch.* p. 1419 (2022)
43. Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: Linear-time and field-agnostic snarks for R1CS. In: *CRYPTO (2)*. *Lecture Notes in Computer Science*, vol. 14082, pp. 193–226. Springer (2023)
44. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: *EUROCRYPT (2)*. *Lecture Notes in Computer Science*, vol. 9057, pp. 253–280. Springer (2015)
45. Haböck, U.: Multivariate lookups based on logarithmic derivatives. *IACR Cryptol. ePrint Arch.* p. 1530 (2022)
46. Heath, D., Kolesnikov, V.: Stacked garbling for disjunctive zero-knowledge proofs. In: *EUROCRYPT (3)*. *Lecture Notes in Computer Science*, vol. 12107, pp. 569–598. Springer (2020)
47. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: *ASIACRYPT*. *Lecture Notes in Computer Science*, vol. 6477, pp. 177–194. Springer (2010)
48. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: *STOC*. pp. 723–732. ACM (1992)
49. Kosba, A.E., Papadopoulos, D., Papamanthou, C., Song, D.: MIRAGE: succinct arguments for randomized algorithms with applications to universal zk-snarks. In: *USENIX Security Symposium*. pp. 2129–2146. USENIX Association (2020)
50. Kothapalli, A., Setty, S.: Supernova: Proving universal machine executions without universal circuits. *IACR Cryptol. ePrint Arch.* p. 1758 (2022)
51. Kothapalli, A., Setty, S., Tzialla, I.: Nova: Recursive zero-knowledge arguments from folding schemes. In: *CRYPTO (4)*. *Lecture Notes in Computer Science*, vol. 13510, pp. 359–388. Springer (2022)
52. Kothapalli, A., Setty, S.T.V.: Cyclefold: Folding-scheme-based recursive arguments over a cycle of elliptic curves. *IACR Cryptol. ePrint Arch.* p. 1192 (2023)
53. Lee, J., Nikitin, K., Setty, S.T.V.: Replicated state machines without replicated execution. In: *IEEE Symposium on Security and Privacy*. pp. 119–134. IEEE (2020)
54. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. *J. ACM* **39**(4), 859–868 (1992)

55. Micali, S.: CS proofs (extended abstracts). In: FOCS. pp. 436–453. IEEE Computer Society (1994)
56. Nguyen, W.D., Boneh, D., Setty, S.T.V.: Revisiting the nova proof system on a cycle of curves. In: AFT. LIPIcs, vol. 282, pp. 18:1–18:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023)
57. Nguyen, W.D., Datta, T., Chen, B., Tyagi, N., Boneh, D.: Mangrove: A scalable framework for folding-based snarks. IACR Cryptol. ePrint Arch. p. 416 (2024)
58. Posen, J., Kattis, A.A.: Caulk+: Table-independent lookup arguments. IACR Cryptol. ePrint Arch. p. 957 (2022)
59. Setty, S.T.V.: Spartan: Efficient and general-purpose zk snarks without trusted setup. In: CRYPTO (3). Lecture Notes in Computer Science, vol. 12172, pp. 704–737. Springer (2020)
60. Setty, S.T.V., Angel, S., Gupta, T., Lee, J.: Proving the correct execution of concurrent services in zero-knowledge. In: OSDI. pp. 339–356. USENIX Association (2018)
61. Setty, S.T.V., Thaler, J., Wahby, R.S.: Customizable constraint systems for succinct arguments. IACR Cryptol. ePrint Arch. p. 552 (2023)
62. Setty, S.T.V., Thaler, J., Wahby, R.S.: Unlocking the lookup singularity with lasso. IACR Cryptol. ePrint Arch. p. 1216 (2023)
63. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: TCC. Lecture Notes in Computer Science, vol. 4948, pp. 1–18. Springer (2008)
64. Wahby, R.S., Setty, S.T.V., Ren, Z., Blumberg, A.J., Walfish, M.: Efficient RAM and control flow in verifiable outsourced computation. In: NDSS. The Internet Society (2015)
65. Xie, T., Zhang, Y., Song, D.: Orion: Zero knowledge proof with linear prover time. In: CRYPTO (4). Lecture Notes in Computer Science, vol. 13510, pp. 299–328. Springer (2022)
66. Xiong, A.L., Chen, B., Zhang, Z., Bünz, B., Fisch, B., Krell, F., Camacho, P.: VERI-ZEXE: decentralized private computation with universal setup. IACR Cryptol. ePrint Arch. p. 802 (2022)
67. Yang, Y., Heath, D.: Two shuffles make a RAM: improved constant overhead zero knowledge RAM. IACR Cryptol. ePrint Arch. p. 1115 (2023)
68. Yang, Y., Heath, D., Hazay, C., Kolesnikov, V., Venkatasubramaniam, M.: Batchman and robin: Batched and non-batched branching for interactive ZK. In: CCS. pp. 1452–1466. ACM (2023)
69. Yang, Y., Heath, D., Hazay, C., Kolesnikov, V., Venkatasubramaniam, M.: Tight zk cpu: Batched zk branching with cost proportional to evaluated instruction. IACR Cryptol. ePrint Arch. p. 456 (2024)
70. Zapico, A., Buterin, V., Khovratovich, D., Maller, M., Nitulescu, A., Simkin, M.: Caulk: Lookup arguments in sublinear time. In: CCS. pp. 3121–3134. ACM (2022)
71. Zapico, A., Gabizon, A., Khovratovich, D., Maller, M., Ràfols, C.: Baloo: Nearly optimal lookup arguments. IACR Cryptol. ePrint Arch. p. 1565 (2022)
72. Zhang, Y., Genkin, D., Katz, J., Papadopoulos, D., Papamanthou, C.: vram: Faster verifiable RAM with program-independent preprocessing. In: IEEE Symposium on Security and Privacy. pp. 908–925. IEEE Computer Society (2018)

Verifiable Computation



Proofs for Deep Thought: Accumulation for Large Memories and Deterministic Computations

Benedikt Bünz^(✉)  and Jessica Chen^(✉) 

New York University, New York, USA
{bb,jessicachen}@nyu.edu

Abstract. An important part in proving machine computation is to prove the correctness of the read and write operations performed from the memory, which we term *memory-proving*. Previous methodologies required proving Merkle Tree openings or multi-set hashes, resulting in relatively large proof circuits. We construct an efficient memory-proving Incrementally Verifiable Computation (IVC) scheme from accumulation, which is particularly useful for machine computations with large memories and deterministic steps. In our scheme, the IVC prover P_{IVC} has cost entirely independent of the memory size T and only needs to commit to approximately 15 field elements per read/write operation, marking a more than 100X improvement over prior work. We further reduce this cost by employing a modified, accumulation-friendly version of the GKR protocol. In the optimized version, P_{IVC} only needs to commit to 6 small memory-table elements per read/write. If the table stores 32-bit values, then this is equivalent to committing to less than one single field element per read and write. Our modified GKR protocol is also valuable for proving other deterministic computations within the context of IVC. Our memory-proving protocol can be extended to support key-value stores. The full version of this article can be found online [BC24]

Keywords: Proof system · Accumulation Scheme · Incrementally Verifiable Computation

1 Introduction

Consider the scenario where one or multiple clients outsource a large computation, possibly of infinite steps, to an untrusted server. For example, clients might want to continuously verify that all transactions in a blockchain are valid. Naturally, the clients would like the server to provide a certificate, which would allow the clients to verify that all the computation steps run up to that point were correct and even to continue the computation from that point onwards. The efficiency of verification necessitates that the size of the proof and the complexity of

The full version of this paper is on IACR ePrint at <https://eprint.iacr.org/2024/325>.

© International Association for Cryptologic Research 2025
K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15488, pp. 269–301, 2025.
https://doi.org/10.1007/978-981-96-0935-2_9

its verification be independent of the length of the computation. Moreover, since the computation can be long or even unbounded, it would be ideal if the server can provide the current state and a certificate upon request from the client *at any point*. This is achieved by maintaining a running certificate or proof that can be efficiently updated with each computation step. A system that achieves these properties is called an *incrementally verifiable computation* (IVC) system [Val08]¹.

IVC enables the server/prover to produce an output z_{IVC} , along with a proof π_{IVC} upon request from the client/verifier without requiring *a priori* knowledge of an upper bound on the number of computation steps. With a valid π_{IVC} , a client/verifier can be convinced z_{IVC} is the output of the correct execution of a (potentially non-deterministic) machine computation up to this point, and can even continue the computation. Recent developments have demonstrated that IVC can be constructed from simple public-coin interactive protocols featuring algebraic verifiers, such as protocols where the prover simply sends the witness. This is achieved through the use of accumulation² or folding schemes [BGH19, BCMS20, BCLMS21, BDFG21, KST22, BC23, EG23]. The resulting IVC has essentially the same computational overhead as the accumulation scheme. The cost of the resulting IVC prover depends on two main factors:

1. The size of the recursive circuit, predominantly comprising the accumulation verifier V_{acc} . Since the size of V_{acc} only depends on the algebraic degree of the verifier and the number of rounds in the underlying protocol (rather than the communication or verification complexity), minimizing these two factors are crucial for reducing the cost of the IVC prover.
2. The cost of the accumulation prover P_{acc} , which is mainly influenced by the commitment cost to the prover messages, and thus is dependent on the number of elements in the prover messages of the underlying interactive protocol.

The general paradigm of using IVC to prove machine computations involves first proving the correctness of computation under the assumption that memory accesses were executed correctly, recording all the read/write operations in the circuit, and then proving the correctness of the recorded read/write operations. The primary challenge lies in the latter, i.e. efficiently proving the correctness of memory accesses, which we will refer to as *memory-proving* in this work. With the above-mentioned recent advancements in IVC construction, **we need only to design a public-coin interactive protocol for memory-proving with an algebraic verifier while ensuring that following three parameters remain small: the number of rounds, the verifier degree, and the number of elements in the prover messages** (ideally independent of T). These parameters are the only factors on which the cost of P_{IVC} depends. Then, by

¹ The literary application of IVC is the machine Deep Thought from the Hitchhiker’s Guide to the Galaxy. It computes the answer to the ultimate question of the universe and life over several thousand years. Given the nonsensical answer (42), it would have been helpful to be able to efficiently verify the correctness of the computation.

² We use accumulation to refer to split-accumulation as defined by [BCLMS21].

applying existing accumulation compilers (e.g., the ProtoStar compiler [BC23]) to this interactive protocol, we can obtain an efficient accumulation scheme for memory-proving, and finally derive an efficient memory-proving IVC scheme from accumulation by utilizing existing IVC compilers (e.g., [BCLMS21]).

The most rudimentary method of performing memory-proving involves unrolling the entire memory into a circuit. However, since a circuit is at least as large as its inputs, this circuit would be of size $O(\ell T)$, which is prohibitively large even medium-sized memories. An alternative approach is for the prover to simulate memory-checking internally and prove that the memory accesses would have been accepted by the memory-checking verifier, who only keeps a small local state. In all previous works, using this approach, the prover’s cost is dependent on the memory size T and/or hashing is required within the circuit. For instance, Spice [SAGL18] employed offline memory-checking, requiring approximately 1500 constraints³ per read and write operation. Since the prover needs to transmit at least one proof element per constraint, this results in 1500 elements in the prover message per read/write operation, and thus 1500 commitments per accumulation step for P_{IVC} . In contrast, ProtoStar recently showed a memory-proving protocol for static read-only memory that utilises the LogUp argument [Hab22], in which the prover only performs two group scalar multiplications per read instruction [BC23]. The prover’s cost in ProtoStar is independent of the memory size T and does not involve multi-set hashing. However, their approach does not support writes into a dynamic memory [BC23].

In this paper, we present an interactive protocol for memory-proving inspired by the LogUp argument [Hab22]. Using accumulation techniques, we obtain an IVC scheme for memory-proving with minimal prover overhead. We then show an optimization of our scheme which employs an accumulation-friendly version of the GKR protocol to further reduce the prover overhead. We note that this adapted GKR protocol has other applications beyond improving our memory-proving protocol.

$O(\ell)$ Memory-Proving P_{IVC} ProtoStar has previously demonstrated that LogUp is well suited for accumulation and, thus, IVC. It can be used to verify the existence of a set of witness values in a static table of values [BC23]. We design LogUp-styled arguments to support reading from and writing to a fully dynamic table.

One key challenge we address is proving only $O(\ell)$ table values were altered in a table of size $T \gg \ell$, while ensuring that the cost of P_{IVC} remains independent of T . Our memory-proving protocol is public-coin with an algebraic verifier, featuring 2 rounds of communication⁴, verifier degree 3, and only $O(\ell)$ elements in prover messages, where ℓ denotes the number of reads and writes performed in each computation step. This means it can be turned into an efficient accumulation scheme using existing accumulation compiler (e.g. [BC23]). The resulting

³ In group-based proof systems, the prover typically computes at least one multi-scalar multiplication that is as large as the number of constraints.

⁴ Each round consists of a prover message and is possibly followed by a verifier challenge.

P_{IVC} only needs to commit to $O(\ell)$ elements, which is independent of the memory size T .

This significantly improves on prior work, which either relied on Merkle trees requiring $\log T$ hashes per memory access or required multi-set hashes [SAGL18]. These prior methods are particularly costly in the context of memory-proving, where the hashes result in large proving circuits. In contrast, our resulting memory-proving scheme is practically efficient with the prover only having to commit to 15 field values per memory access. In addition, since the prover cost is completely independent of the memory size T , our protocol can be extended to the setting of key-value store. We describe this extension in detail at the end of Sect. 5.1.

Optimizing Memory-Proving with GKR. One limitation of the scheme is that the prover needs to commit to 6 large field elements per memory access, i.e. each of size λ bits, even if the memory entries themselves are small. This is because in the memory-proving interactive protocol, the prover needs to send 6 vectors consisting of inverses of the form $\frac{1}{r+t_i}$ where r is a constant, and each t_i is a small table entry. To resolve this overhead, we draw inspiration from [STW23, PH23] and compute this sum using formal fractions and a modified GKR protocol. Our modified protocol retains GKR’s ability to prove deterministic layered computations without committing to the intermediate values. Specifically, the protocol relies on bivariate sumcheck instead of multilinear sumcheck, reducing the number of rounds per layer to 3. In the context of the ProtoStar accumulation compiler, this reduction in number of rounds significantly lowers the recursive overhead in IVC.

With the power of GKR, the memory-proving protocol no longer requires computing and committing to the large inverses. Thus, if we read/write ℓ s -bit values from memory, the number of group operations decreases from $O(\lambda\ell)$ to $O(s\ell)$, i.e., the actual size of the data that is read/written. We provide a brief overview of the resulting efficiency of our protocol in Table 1. Most importantly the prover only needs to commit to 6 elements that are as large as the table entries for each read/ write. If the table contains 32-bit entries then this is equivalent to committing to 192bits per read/write or less than a single 256bit field element. We also introduce several optimizations for our GKR-powered memory-proving protocol, which further reduce the number of GKR rounds.

Table 1. Efficiency Table for our Memory-Proving Protocol. T is the memory size, and ℓ is the number of read/write operations. \mathbb{T} is the set of table entries, which might only contain small field elements. See Table 3 for an explanation of the columns and symbols, and more details.

	P_{acc} Time	V_{acc} Time
Plain	$(6\ell, \mathbb{T})\text{-MSM} + (9\ell, \mathbb{F})\text{-MSM}$	$3\mathbb{G}$
Using GKR	$(6\ell, \mathbb{T})\text{-MSM}$	$O(\log T)\mathbb{G}$

IVC for Deterministic Computations. GKR has numerous other applications in accumulation beyond enhancing our memory-proving protocol. In fact, for proving *any* low-depth deterministic computations, GKR only requires committing to the inputs and outputs, not the intermediate values. We demonstrate the utility of this by describing an accumulation-friendly GKR protocol for computing group scalar multiplications, which is the dominant cost within the recursive circuit.

1.1 Related Work

IVC and Accumulation. Valiant [Val08] introduced incrementally verifiable computation (IVC) and showed that IVC can be built from Succinct Non-interactive ARGuments of Knowledge (SNARKs). The core concept involves the prover generating a SNARK at each computation step, certifying both the current step and the verification of the SNARK from the previous step. The latter part is commonly referred to as the *recursive circuit*. Subsequent to Valiant’s work, an important line of research [BCCT13, BCTV14, COS20] has enhanced the practicality of IVC, studied its generalization to arbitrary graphs (Proof-Carrying Data, PCD), and advanced its theoretical foundations.

Halo [BGH19] showed that IVC can be constructed from simpler assumptions, sparking research on accumulation [BCMS20, BCLMS21, BDFG21, KST22, BC23, EG23]. The idea is to construct IVC by simply accumulating or batching the verification of non-interactive arguments, postponing verification to the end of each IVC step. In essence, in each accumulation round, the prover produces a new argument for the current step and proves its correct accumulation into the existing accumulator. The accumulation step can be as straightforward as taking a random linear combination between two vector commitments, and verifying the accumulation step can be significantly cheaper than verifying the proof. The more computationally intensive final verification, which is called the *decision* step in accumulation, is executed only at the end of IVC step to verify the correctness of the accumulated commitment. A valid accumulator implies that all the accumulated proofs were valid.

Recently, ProtoStar introduced a new recipe for constructing accumulation schemes and IVC [BC23] from *any* interactive public-coin protocol Π with an algebraic verifier. The resulting accumulation verifier V_{acc} depends only on the number of rounds and the verifier degree in the underlying interactive protocol Π , and the resulting accumulation prover P_{acc} ’s main cost is committing to all the prover messages in Π . Using the [BCLMS21] compiler, an accumulation scheme for NP directly yields an IVC, where P_{IVC} ’s cost for computing the predicate is proportional to the cost of P_{acc} and the recursive circuit consisting of V_{acc} .

Concurrent work [APPK24] also constructed an accumulation scheme for GKR. However, they utilize the multi-linear version of GKR and batch the polynomial evaluation, similar to [BCMS20].

Memory-Checking and Lookup Arguments. Memory-checking [BEGKN91] enables an untrusted server to convince a client that a set of read/write operations

is consistent with a memory without having to send the entire memory to the client. Each entry of read/write operation consists of an address a , a value v and a timestamp t . If a value v was written to a at time t , then any read at time $t' > t$ from a shall return v with the timestamp t , unless there was another write to a in the meantime. We briefly highlight two constructions and their limitations here, and refer to Appendix B of Jolt [AST23] for an excellent overview of memory-checking techniques.

One approach stores the memory in a Merkle Tree [BFRSBW13, BCTV14]. For every read operation, the prover opens the Merkle Tree at the relevant address. For every write operation, the prover shows that the Merkle Tree is correctly updated. The verification for either step requires $O(\log T)$ hashes, and the prover’s computational work is also $O(\log T)$, where T is the size of the memory. When this technique is used within IVC, the memory-checking verifier is part of the proving circuit, and $\log(T)$ hashes per read and write operation become a significant overhead.

The other common approach, dating back to [BEGKN91] and later refined in [CDvGS03, DNRV09, SAGL18], relies on proving that the constructed sets of reads and writes form a permutation. The state-of-the-art work Spice [SAGL18] employs multi-set hashes and proves that the hash was evaluated correctly, which results in over 1500 constraints per read/write operation, two orders of magnitude more than our approach. The approach also requires a linear scan of the memory at the end of the computation, but similar as in our construction this can be deferred to a final decider.

Recently, there has been increased attention to a related primitive called lookup arguments. Lookup arguments can be used to verify read operation in a static, possibly preprocessed memory. A recent line of work [ZBKMNS22, PK22, GK22, ZGKMR22, EFG22] showed that in the preprocessing setting, one can achieve lookup arguments independent of the table size and quasi-linear in the number of read operations. Lasso [STW23] improves on these ideas by enabling a fully linear prover and independence of the table size for structured table. In the context of IVC, ProtoStar [BC23] gave a lookup argument based on LogUp [Hab22] that is independent of the table size (for arbitrary tables) and only requires two group scalar multiplications per read. Unfortunately, all these lookup arguments only work for static tables and read operations. We construct a memory-proving argument (which is more general than a lookup) that is still independent of the table size and has minimal overhead.

1.2 Technical Overview

Our construction heavily relies on the ProtoStar compiler [BC23], which we describe in Theorem 1 in Sect. 2.5. It gives a recipe for constructing accumulation schemes and IVC [BC23] from *any* interactive public-coin protocol with an algebraic verifier. We summarize the recipe here into five steps:

1. Begin with *any* k -round interactive public-coin protocol featuring L verification checks of maximum degree d , and prover messages comprising n nonzero elements.

2. Compress the communication by using a homomorphic vector commitment (e.g. Pedersen commitment) to commit to each vector in the prover messages.
3. Make the protocol non-interactive through the Fiat-Shamir transformation.
4. Use the ProtoStar compiler to convert the non-interactive protocol into an accumulation scheme. The accumulation scheme combines the current argument with an accumulator (which has the same form as the argument) by taking a random linear combination of the committed prover messages with the accumulator messages. It also computes a new verification equation by appropriately canceling out the cross error terms resulted from the accumulation.
5. If the underlying protocol can prove NP-complete relations, such as circuits, then the [BCLMS21] IVC compiler can be applied to construct an IVC scheme from the accumulation scheme for any function F . The compiler ensures the correct execution of the accumulation verifier alongside proving F .

Following this recipe, we design special-sound, algebraic protocols for memory-proving and GKR. One important design goal is to keep the complexity of the accumulation verifier V_{acc} low, as V_{acc} is the dominant component in the recursive circuit. Notably, the complexity of V_{acc} relies solely on k and d , without any dependence on n or L whatsoever. Another design goal is to minimize the commitment cost of the accumulation prover P_{acc} , which is directly contingent on the number of nonzero elements in prover messages, as committing to 0 is free in Pedersen commitment. Therefore, to leverage the ProtoStar compiler to design an efficient IVC scheme where P_{IVC} cost is independent of the memory size T , **we need to design an interactive, algebraic memory-proving protocol with small values for number of rounds k , verifier degree d and number of nonzero elements in prover messages n .** This implies that n should be independent of T , since otherwise the cost of P_{acc} will be $O(T)$ even if the number of memory accesses is much smaller than T .

Constructing Read List and Write List. We assume the list of “reads” and the list of “writes” were constructed similarly to the way in the classic offline memory-checking process [BEGKN91, CDvGS03, SAGL18]. Each entry in the lists is in the form of a tuple (address, value, timestamp), with the local timestamp incremented after each write operation. The specific construction is described in Sect. 3.

If all memory accesses were performed correctly, the constructed lists should satisfy three properties: 1) the read list and the write list should be permutations of each other; 2) the initial reads are consistent with the starting/old memory; and 3) the new memory is updated only at the addresses written to and with the correct amount. Note that the third memory-update (or mem-update for short) property requires examining all T addresses, not only the ones touched by memory accesses but also the ones untouched. Therefore, the main challenge in designing an efficient memory-proving protocol lies in proving correct mem-update in time independent of T .

LogUp Based Mem-Update. The starting point of our construction is the *LogUp* lookup argument [Hab22] which uses the fact that the set of values in $\mathbf{w} = [\mathbf{w}_i]_{i=1}^\ell$ is contained in a table $\mathbf{t} = [\mathbf{t}_i]_{i=1}^T$ if and only if

$$\sum_{j=1}^{\ell} \frac{1}{X + \mathbf{w}_j} = \sum_{i=1}^T \frac{\mathbf{m}_i}{X + \mathbf{t}_i},$$

where \mathbf{m}_i is the multiplicity of \mathbf{t}_i in \mathbf{w} for every $i \in [T]$ and X is a random variable. ProtoStar [BC23] showed that the LogUp argument can be efficiently accumulated. Importantly, it observes that the prover messages in the protocol for LogUp argument, e.g. $\mathbf{m} = [\mathbf{m}_i]_{i=1}^T$, only contains ℓ nonzero entries. This means, in the context of the ProtoStar compiler and IVC, the accumulation prover P_{acc} and thus the IVC prover P_{IVC} only needs to do $O(\ell)$ work. However, the LogUp argument only supports read operations and not write operations.

We attempt to modify the LogUp argument to use it for mem-update. Assume, \mathbf{w} corresponds to the ℓ -sized update vector (the difference between the final written value and the initial read value from each address), and \mathbf{t} corresponds to the T -sized vector Δ that represents the difference between the new memory and the old memory, i.e. $\Delta := \text{NM} - \text{OM}$. However, the LogUp argument only cares about the membership of the \mathbf{w} values but not their positions in Δ ; in other words, the argument only indicates that some memory value is changed by \mathbf{w}_j , but does not constrain the change to any specific address. In addition, it is not immediately clear how to update the memory or compute the right hand side with Δ in a manner that does not require a linear scan.

To resolve the first issue, we add the address vector to random linear combination in the denominators. That is,

$$\sum_{j=1}^{\ell} \frac{1}{X + Y \cdot \mathbf{b}_j + \mathbf{w}_j} = \sum_{i=1}^T \frac{\mathbf{m}_i}{X + Y \cdot i + \Delta_i}$$

holds if and only if $\mathbf{w}_j = \Delta_{\mathbf{b}_j}$ for every $j \in [\ell]$, where $\mathbf{b} = [\mathbf{b}_j]_{j=1}^\ell$ is an address vector and Y is another random variable. Note that this is an *indexed* LogUp argument where we ensure not only the membership of the values but also their precise indices in the lookup table. In this indexed lookup argument, \mathbf{m}_i only takes on the values 0 or 1. Still, this modified lookup argument is not sufficient, as it does not ensure that Δ is 0 at the positions for which there had been no read or write operation. This is an important criteria for correct mem-update, since an adversarial prover may use non-zero values in Δ to change the memory state arbitrarily.

We make a key observation that the correct Δ should simply be a T -sized sparse representation of \mathbf{w} . To ensure that Δ is 0 at unmodified addresses, we set the numerators to \mathbf{w}_j, Δ_i instead of 1, \mathbf{m}_i . Namely,

$$\sum_{j=1}^{\ell} \frac{\mathbf{w}_j}{X + Y \cdot \mathbf{b}_j + \mathbf{w}_j} = \sum_{i=1}^T \frac{\Delta_i}{X + Y \cdot i + \Delta_i}$$

Note that the i th fraction on the right hand side is 0 if and only if the i th value is unmodified by any write operation, and is equal to the left hand side if and only if the i th value is modified by the correct amount. Only ℓ out of all T fractions on the right hand side are nonzero, which implies an honest prover only need to do $O(\ell)$ work, resolving the second issue aforementioned. Section 4.3 shows that this LogUp-style mem-update argument is secure and indeed leads to a protocol with prover complexity independent of T .

LogUp Powered Memory-Proving. The mem-update argument can be used to show that the memory is updated strictly at the addresses written to and with the correct amount. We can then use a homomorphic commitment to Δ to efficiently update our commitment to the memory state. In addition, we can use the indexed LogUp argument demonstrated in the intermediate step above to show that all the values initially read are consistent with the old memory. Nevertheless, merely checking these two properties (property 2 and 3) only suffice in a system where all write operations happen synchronously at the end of the computation step. Without additional checks, we would need to first update the memory whenever we read from an address that has been previously written to. This requires an expensive homomorphic commitment operation to be executed by the verifier as part of the recursive circuit. To resolve this we employ the classic permutation-based offline memory-checking idea [BEGKN91] and add a check for property 1 in our memory-proving protocol.

All three subprotocols are based on the LogUp argument and are described in Sect. 4. Section 5 discusses the overall memory-proving protocol and its efficiency.

Accumulation-Friendly Verison of GKR. Our memory-proving protocols have *almost* optimal parameters. It requires committing⁵ to only 15 field elements per memory access. However, 9 of these field elements consist of large field elements, i.e. $\log|\mathbb{F}|$ -bit, even if the memory itself only consists of small entries. For instance, say the memory only contains 32-bit entries; using homomorphic commitments requires fields of size at least 2^{256} , which is a factor 8 blowup. Concretely, in this example the 9 large elements contribute 2300 bits and the 6 small elements only 192 bits to the prover’s commitment cost.

Removing this blowup motivates the second orthogonal but highly compatible contribution of this paper: We construct an efficient accumulation scheme for the GKR protocol. GKR can be used to prove low-depth deterministic computations while only committing to the computation’s inputs and outputs but not the intermediate values. Note that GKR is a special-sound interactive protocol with an algebraic verifier, which means it can directly be compiled with the ProtoStar compiler to an accumulation scheme. Unfortunately, GKR has $O(k \cdot \log n)$ rounds where k is the depth of the circuit and n its width. Compilation results

⁵ Committing is by far the dominant prover cost in these systems. Committing to a message is between 100 and 1000 times as expensive as doing field operations on the same message. See <https://zka.lc/>.

in an accumulation verifier with $k \cdot \log n$ group scalar multiplications. In the context of IVC, the accumulation verifier becomes part of the recursive circuit, and this is a significant overhead, especially when compared with other accumulation schemes which only have 1 to 3 group operations [KST22, KS23, BC23]. Our goal is, therefore, to reduce the number of rounds of GKR while maintaining the attractive efficiency properties and the compatibility with the ProtoStar compiler.

In every round, GKR runs a multivariate sumcheck protocol, which has $\log n$ rounds. As a strawman, we can replace this multivariate sumcheck with a univariate one. This immediately reduces the number of GKR rounds from $k \cdot \log n$ to just k . Univariate sumcheck requires sending a quotient polynomial that is as large as the domain of the sumcheck, in our case $O(n)$. Committing to this polynomial would be at least as expensive as directly committing to the intermediate wires of the circuit, thus removing the benefit of using GKR. Fortunately, the idea of using a higher degree sumcheck with fewer variables can still help. Moving to a bivariate sumcheck reduces the communication to $O(\sqrt{n})$ while being only a 3-round protocol. The $O(\sqrt{n})$ commitment cost is, in most applications, dominated by the cost of committing to the input and output layers; even if not, we show that one can use a c -variate sumcheck to ensure that the sumchecks commitment cost is marginal. Using a bivariate sumcheck presents us with a couple of challenges. First, the verifier needs to evaluate a $O(\sqrt{n})$ degree polynomial, which is a $O(\sqrt{n})$ degree check if done naively. To resolve this we built a polynomial evaluation protocol, where with aid from the prover, the verification degree reduces to merely 3, independent of the degree of the polynomial.

Additionally, GKR batches polynomial evaluations, after each sumcheck, in order to only evaluate the next layer at a single point. In bivariate sumcheck, this would require computing a high-degree interpolation polynomial. We show that it is much simpler and more efficient to directly batch the resulting sumchecks. This observation is also applicable to multivariate sumchecks. We then construct a specific GKR protocol for computing the sum of fractions, e.g. $\sum_{i=1}^n \frac{n_i}{d_i}$, similar to [PH23]. We also give specific optimizations for this instantiation, such as breaking up the circuit into multiple parts, while still maintaining the asymptotic properties. This optimization takes advantage of the circuit structure of sums of fractions, where the number of sums halves in every layer.

1.3 Roadmap

In Sect. 2, we provide the necessary preliminaries to comprehend our construction. We describe the desired construction for lists of read/write operations in Sect. 3 and outline three properties that consistent read/write lists should uphold. Subsequently, in Sect. 4, we introduce three LogUp-style special-sound subprotocols, each tailored for proving one of the three aforementioned properties. These subprotocols are combined in parallel to form the memory-proving interactive protocol Π_{MP} in Sect. 5, which exhibits the desired characteristics for conversion into an efficient accumulation scheme and IVC using the ProtoStar

compiler. Specifically, Π_{MP} has only 2 rounds and verifier degree 3, and its number of nonzero elements in prover messages is independent of T . In Sect. 7, we elucidate how our accumulation-friendly version of GKR (components described in Sect. 6) can be leveraged to optimize our memory-proving IVC scheme, and we highlight several other useful applications of GKR in the context of IVC. The proofs, the extension of our memory-proving protocol to the setting of key-value store, and other details are deferred to the full version of this paper [BC24] due to lack of space.

2 Preliminaries

Notation. For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \dots, n\}$. We denote λ as the security parameter and use \mathbb{F} to denote a field of prime order p such that $\log(p) = \Omega(\lambda)$. For list of tuples $ltup = [(a_i, b_i, c_i, \dots)]_{i=1}^k$ of arbitrary length k , we use $ltup.\mathbf{a}$ to denote the list $[a_i]_{i=1}^k$, and $ltup.(\mathbf{a}, \mathbf{b})$ to denote the list $[(a_i, b_i)]_{i=1}^k$. For function f , \tilde{f} denotes the bivariate extension of f .

2.1 Special-Sound Protocols

We take the definition of special-soundness from [AFK22, BC23].

Definition 1 (Public-coin interactive proof). *An interactive proof $\Pi = (\text{P}, \text{V})$ for relation \mathcal{R} is an interactive protocol between two probabilistic machines, a prover P , and a polynomial time verifier V . Both P and V take as public input a statement pi and, additionally, P takes as private input a witness $\mathbf{w} \in \mathcal{R}(\text{pi})$. The verifier V outputs 0 if it accepts and a non-zero value otherwise. Its output is denoted by $(\text{P}(\mathbf{w}), \text{V})(\text{pi})$. Accordingly, we say the corresponding transcript (i.e., the set of all messages exchanged in the protocol execution) is accepting or rejecting. The protocol is public coin if the verifier randomness is public. The verifier messages are referred to as challenges. Π is a $(2k - 1)$ -move protocol if there are k prover messages and $k - 1$ verifier messages.*

Definition 2 (Tree of transcript). *Let $\mu \in \mathbb{N}$ and $(a_1, \dots, a_\mu) \in \mathbb{N}^\mu$. An (a_1, \dots, a_μ) -tree of transcript for a $(2\mu + 1)$ -move public-coin interactive proof Π is a set of $a_1 a_2 \dots a_\mu$ accepting transcripts arranged in a tree of depth μ and arity a_1, \dots, a_μ respectively. The nodes in the tree correspond to the prover messages and the edges to the verifier's challenges. Every internal node at depth $i - 1$ ($1 \leq i \leq \mu$) has a_i children with distinct challenges. Every transcript corresponds to one path from the root to a leaf node. We simply write the transcripts as an (a^μ) -tree of transcript when $a = a_1 = a_2 = \dots = a_\mu$.*

Definition 3 (Special-sound Interactive Protocol). *Let $\mu, N \in \mathbb{N}$ and $(a_1, \dots, a_\mu) \in \mathbb{N}^\mu$. A $(2\mu + 1)$ -move public-coin interactive proof Π for relation \mathcal{R} where the verifier samples its challenges from a set of size N is (a_1, \dots, a_μ) -out-of- N special-sound if there exists a polynomial time algorithm that, on input pi and any (a_1, \dots, a_μ) -tree of transcript for Π outputs $\mathbf{w} \in \mathcal{R}(\text{pi})$. We simply denote the protocol as an a^μ -out-of- N (or a^μ) special-sound protocol if $a = a_1 = a_2 = \dots = a_\mu$.*

2.2 Commitment Scheme

Definition 4 (Commitment Scheme). (Definition 6 from [BC23]) $\text{cm} = (\text{Setup}, \text{Commit})$ is a binding commitment scheme, consisting of two algorithms: $\text{Setup}(\lambda) \rightarrow \text{ck}$ takes as input the security parameter and outputs a commitment key ck .

$\text{Commit}(\text{ck}, \mathbf{m} \in \mathcal{M}) \rightarrow C \in \mathcal{C}$, takes as input the commitment key ck and a message \mathbf{m} in \mathcal{M} and outputs a commitment $C \in \mathcal{C}$.

The scheme is binding if for all polynomial-time randomized algorithms P^* :

$$\Pr \left[\begin{array}{c} \text{Commit}(\text{ck}, \mathbf{m}) = \text{Commit}(\text{ck}, \mathbf{m}') \\ \wedge \\ \mathbf{m} \neq \mathbf{m}' \end{array} \middle| \begin{array}{c} \text{ck} \leftarrow \text{Setup}(1^\lambda) \\ \mathbf{m}, \mathbf{m}' \leftarrow \mathsf{P}^*(\text{ck}) \end{array} \right] = \text{negl}(\lambda)$$

Homomorphic commitment. (Adapted from Definition 17 in [KST22]) Let $(\mathcal{C}, +)$ be an additive group of prime order p . We say the commitment is homomorphic if for all commitment key produced from $\text{Setup}(1^\lambda)$, and for any $\mathbf{m}_1, \mathbf{m}_2 \in \mathcal{M}^2$, $\text{Commit}(\text{ck}, \mathbf{m}_1) + \text{Commit}(\text{ck}, \mathbf{m}_2) = \text{Commit}(\text{ck}, \mathbf{m}_1 + \mathbf{m}_2)$.

2.3 Lookup Relation

Definition 5. (Definition 12 of [BC23]) Given configuration $\mathcal{C}_{LK} := (T, \ell, \mathbf{t})$ where ℓ is the number of lookups and $\mathbf{t} \in \mathbb{F}^T$ is the lookup table, the relation \mathcal{R}_{LK} is the set of tuples $\mathbf{w} \in \mathbb{F}^\ell$ such that $\mathbf{w}_i \in \mathbf{t}$ for all $i \in [\ell]$.

Lemma 1. (Lemma 5 of [Hab22])⁶ Let \mathbb{F} be a field of characteristic $p > \max(\ell, T)$. Given two sequences of field elements $[\mathbf{w}_i]_{i=1}^\ell$ and $[\mathbf{t}_i]_{i=1}^T$, we have $\{\mathbf{w}_i\} \subseteq \{\mathbf{t}_i\}$ as sets (with multiples of values removed) if and only if there exists a sequence $[\mathbf{m}_i]_{i=1}^T$ of field elements such that

$$\sum_{i=1}^{\ell} \frac{1}{X + \mathbf{w}_i} = \sum_{i=1}^T \frac{\mathbf{m}_i}{X + \mathbf{t}_i}. \tag{1}$$

2.4 Vector-Valued Lookup

Definition 6. (Definition 13 in [BC23]) Consider configuration $\mathcal{C}_{VLK} := (T, \ell, v \in \mathbb{N}, \mathbf{t})$ where ℓ is the number of lookups, and $\mathbf{t} \in (\mathbb{F}^v)^T$ is a lookup table in which the i th ($1 \leq i \leq T$) entry is $\mathbf{t}_i := (\mathbf{t}_{i,1}, \dots, \mathbf{t}_{i,v}) \in \mathbb{F}^v$. A sequence of vectors $\mathbf{w} \in (\mathbb{F}^v)^\ell$ is in relation \mathcal{R}_{VLK} if and only if for all $i \in [\ell]$, $\mathbf{w}_i := (\mathbf{w}_{i,1}, \dots, \mathbf{w}_{i,v}) \in \mathbf{t}$.

As noted in Sect. 3.4 of [Hab22], we can extend Lemma 1 and replace (1) with

$$\sum_{i=1}^{\ell} \frac{1}{X + w_i(Y)} = \sum_{i=1}^T \frac{\mathbf{m}_i}{X + t_i(Y)} \tag{2}$$

⁶ This lookup argument is unofficially referred to as *LogUp*.

where the polynomials are defined as

$$w_i(Y) := \sum_{j=1}^v \mathbf{w}_{i,j} \cdot Y^{j-1}, \quad t_i(Y) := \sum_{j=1}^v \mathbf{t}_{i,j} \cdot Y^{j-1},$$

which represent the witness vector $\mathbf{w}_i \in \mathbb{F}^v$ and the table vector $\mathbf{t}_i \in \mathbb{F}^v$.

2.5 Incremental Verifiable Computation (IVC)

Definition 7 (IVC). (Adapted Definition 5 from [KST22]) An incrementally verifiable computation (IVC) scheme is defined by PPT algorithms (G, P, V) and deterministic K denoting the generator, the prover, the verifier, and the encoder respectively. An IVC scheme (G, K, P, V) satisfies perfect completeness if for any adversary \mathcal{A}

$$\Pr \left[\mathbf{V}(\mathbf{vk}, i, z_0, z_i, \Pi_i) = 0 \left| \begin{array}{l} \mathbf{pp} \leftarrow G(1^\lambda), \\ F, (i, z_0, z_i, z_{i-1}, \omega_{i-1}, \Pi_{i-1}) \leftarrow \mathcal{A}(\mathbf{pp}), \\ (\mathbf{pk}, \mathbf{vk}) \leftarrow K(\mathbf{pp}, F), \\ z_i = F(z_{i-1}, \omega_{i-1}), \\ \mathbf{V}(\mathbf{vk}, i-1, z_0, z_{i-1}, \Pi_{i-1}) = 0, \\ \Pi_i \leftarrow P(\mathbf{pk}, i, z_0, z_i; z_{i-1}, \omega_{i-1}, \Pi_{i-1}) \end{array} \right. \right] = 1$$

where F is a polynomial time computable function. Likewise, an IVC scheme satisfies knowledge soundness if for any constant $n \in \mathbb{N}$, and for all expected polynomial time adversaries P^* , there exists an expected polynomial-time extractor \mathcal{E} such that

$$\Pr_r \left[\begin{array}{l} z_n = z \text{ where} \\ z_{i+1} \leftarrow F(z_i, \omega_i) \\ \forall i \in \{0, \dots, n-1\} \end{array} \left| \begin{array}{l} \mathbf{pp} \leftarrow G(1^\lambda), \\ (F, (z_0, z), \Pi) \leftarrow P^*(\mathbf{pp}, r), \\ (\omega_0, \dots, \omega_{n-1}) \leftarrow \mathcal{E}(\mathbf{pp}, r) \end{array} \right. \right] \approx \\ \Pr_r \left[\mathbf{V}(\mathbf{vk}, (n, z_0, z), \Pi) = 0 \left| \begin{array}{l} \mathbf{pp} \leftarrow G(1^\lambda), \\ (F, (z_0, z), \Pi) \leftarrow P^*(\mathbf{pp}, r), \\ (\mathbf{pk}, \mathbf{vk}) \leftarrow K(\mathbf{pp}, F) \end{array} \right. \right]$$

where r denotes an arbitrarily long random tape.

An IVC scheme satisfies succinctness if the size of the IVC proof Π does not grow with the number of applications n .

Definition 8 (Fiat-Shamir Heuristic). (Definition 9 from [BC23]) The Fiat-Shamir Heuristic, relative to a secure cryptographic hash function H , states that a random oracle NARK with negligible knowledge error yields a NARK that has negligible knowledge error in the standard (CRS) model if the random oracle is replaced with H .

Theorem 1 (ProtoStar compiler). (Theorem 3 from [BC23]) Let \mathbb{F} be a finite field, such that $|\mathbb{F}| \geq 2^\lambda$ and $\text{cm} = (\text{Setup}, \text{Commit})$ be a binding homomorphic commitment scheme for vectors in \mathbb{F} . Let $\Pi_{\text{sps}} = (P_{\text{sps}}, V_{\text{sps}})$ be a special-sound protocol for an NP-complete relation \mathcal{R}_{NP} with the following properties:

- It's $(2k - 1)$ move.
- It's (a_1, \dots, a_{k-1}) -out-of- $|\mathbb{F}|$ special-sound. Such that the knowledge error $\kappa = 1 - \prod_{i=1}^{k-1} (1 - \frac{a_i}{|\mathbb{F}|}) = \text{negl}(\lambda)$
- The inputs are in $\mathbb{F}^{\ell_{\text{in}}}$
- The verifier is degree $d = \text{poly}(\lambda)$ with output in \mathbb{F}^ℓ

Then, under the Fiat-Shamir heuristic for a cryptographic hash function H (Definition 8), there exist two IVC schemes $\text{IVC} = (\text{P}_{\text{IVC}}, \text{V}_{\text{IVC}})$ and $\text{IVC}_{\text{CV}} = (\text{P}_{\text{CV,IVC}}, \text{V}_{\text{CV,IVC}})$ with predicates expressed in \mathcal{R}_{NP} with the efficiencies shown in Table 2.

Table 2. Efficiency of IVC schemes compiled from sps protocol

P_{IVC} native	P_{IVC} recursive	V_{IVC}	$ \pi_{\text{IVC}} $
$\sum_{i=1}^k \mathbf{m}_i^* \mathbb{G}$ $\text{P}_{\text{sps}} + L'(\text{V}_{\text{sps}}, d + 2)$	$k + 2\mathbb{G}$ $k + \ell_{\text{in}} + d + 1\mathbb{F}$ $(k + d + O(1))H + 1H_{\text{in}}$	$\sum_{i=1}^k \mathbf{m}_i \mathbb{G}$ $O(\ell) + \text{V}_{\text{sps}}$	$k + \ell_{\text{in}} + 1\mathbb{F}$ $k + 2\mathbb{G}$ $\sum_{i=1}^k \mathbf{m}_i $

In Table 2, $|\mathbf{m}_i|$ denotes the prover message length; $|\mathbf{m}_i^*|$ is the number of non-zero elements in \mathbf{m}_i ; \mathbb{G} for rows 1-3 is the total length of the messages committed using Commit. \mathbb{F} are field operations. H denotes the total input length to a cryptographic hash, and H_{in} is the hash to the public input and accumulator instance. P_{sps} (and V_{sps}) is the cost of running the prover (and the algebraic verifier) of the special-sound protocol, respectively. $L'(\text{V}_{\text{sps}}, d + 2)$ is the cost of computing the coefficients of the degree $d + 2$ polynomial

$$\begin{aligned}
 e(X) := & \sum_{a=0}^{\sqrt{\ell}-1} \sum_{b=0}^{\sqrt{\ell}-1} (X \cdot \pi \cdot \beta_a + \text{acc} \cdot \beta_a)(X \cdot \pi \cdot \beta'_b + \text{acc} \cdot \beta'_b) \\
 & \sum_{j=0}^d (\mu + X)^{d-j} \cdot f_{j, a+b\sqrt{\ell}}^{\text{V}_{\text{sps}}}(\text{acc} + X \cdot \pi),
 \end{aligned} \tag{3}$$

where all inputs are linear functions in a formal variable X^7 , and $f_{j,i}^{\text{V}_{\text{sps}}}$ is the i th ($0 \leq i \leq \ell - 1$) component of $f_j^{\text{V}_{\text{sps}}}$'s output. For the proof size, \mathbb{G} and \mathbb{F} are the number of commitments and field elements, respectively.

3 Constructing Read List and Write List

In our memory-proving algorithm, we assume that the list of “reads” and the list of “writes” we are given were constructed in a similar way as in the classic

⁷ For example if $f_d = \prod_{i=1}^d (a_i + b_i \cdot X)$ then a naive algorithm takes $O(d^2)$ time but using FFTs it can be computed in time $O(d \log^2 d)$ [CBBZ22].

offline memory-checking process [BEGKN91, CDvGS03, SAGL18]. More importantly, our algorithm makes specific use of the “initial reads” and “final writes” in the memory-checking process, which we explicitly define in this section.

Consider an untrusted server who performs read/write operations to a memory. The memory is represented as a T -sized vector of memory values, where the addresses are the indices $1, \dots, T$. Suppose OM is the starting, old memory. The server locally initializes two lists, RL and WL, to empty lists. As in [BEGKN91], we assume both a value and a discrete timestamp of when the value was written are stored at each memory address. The local timestamp t^* is incremented when some write operation takes place on the data structure.

When a read operation from address a is performed, and the memory responds with a value-timestamp pair (v, t) , the checker updates its local state as follows:

```

checks that  $t^* > t$ 
append  $(a, v, t)$  to RL
stores  $(v, t^*)$  at the memory
append  $(a, v, t^*)$  to WL
 $t^* \leftarrow t^* + 1$ 
    
```

When a write operation of value v' to address a occurs, the checker updates RL, WL in the same way except that the entry appended to WL will contain the new value v' .

Then, we extract the “initial reads” R from RL, and “final writes” W from WL as following:

```

 $R, W, A_R, A_W \leftarrow \{\}$ 
for  $(a, v, t) \in \text{RL}$  do
    if  $a \notin A_R$  then do
        append  $(a, v, t)$  to  $R$ 
         $A_R \leftarrow A_R \cup \{a\}$ 
for  $(a, v, t) \in \text{WL.rev}$  do
    if  $a \notin A_W$  then do
        append  $(a, v, t)$  to  $W$ 
         $A_W \leftarrow A_W \cup \{a\}$ 
sort  $R, W$  by ascending  $a$ 
    
```

At a high level, for each address a , we add the tuple containing a in RL with the smallest timestamp to R , and add the tuple containing a in WL with the largest timestamp to W , and hence the name “initial reads” and “final writes.” Since the entries in RL and WL would be sorted in increasing order of timestamp due to the way they were constructed, traverse the tuples in RL in their natural order, but traverse the tuples in WL backwards (i.e. in descending

order of timestamp), which is what WL.rev indicates in the pseudocode. Finally, we sort R, W by addresses⁸, and return $\text{Rd} := \text{RL} \parallel W$ and $\text{Wr} := \text{WL} \parallel R$.

Lemma 2. (Contrapositive of Lemma 1 from [BEGKN91]) *If Rd and Wr are permutations of each other, then the read/write operations are consistent with each other. In other words, for every address, the value and timestamp read are consistent with the value and timestamp previously written.*

Remark 1. The protocol guarantees that $|\text{RL}| = |\text{WL}|$ and $\text{RL.a} = \text{WL.a}$ if the memory functions correctly. It is therefore clear that if Rd and Wr were to be permutations of each other, then it must be $|W| = |R|$, and $W.a, R.a$ are equal as sets.

Let $\ell := |R| = |W|$ and $k := |\text{RL}| = |\text{WL}|$. Note that k is at most 2ℓ , therefore $k = O(\ell)$.

Remark 2. The memory accesses and the memory updates were performed correctly if and only if the following three properties are satisfied:

1. Rd and Wr are permutations of each other, as described in Lemma 2
2. All the initially read values $\mathbf{R.v}$ are consistent with the old memory OM .
3. The new memory NM is updated only at the addresses written to and with the correct amount. In other words, the T -sized vector $\text{NM} - \text{OM}$ should be an ℓ sparse representation of the ℓ -sized vector $W - R$.

4 Special-Sound Subprotocols for Memory-Proving

We introduce the three LogUp-style subprotocols that will be combined later to build the Read/Write Memory-proving algorithm.

Handling Tuples. For simplicity, we describe the protocols as lookups and permutations on vectors of single values. However, when applied to memory-checking the entries might be tuples of addresses, values, and/or timestamps. Fortunately, this can be handled using a simple random linear combination, akin to the transformation from vector lookups to lookups (Lemma 6 of [BC23]). For sequence $\mathbf{b} = [\mathbf{b}_i]_{i=1}^n$ where each entry is a tuple of $k > 1$ element (i.e. $\mathbf{b}_i = (\mathbf{b}_{(i,j)})_{j=1}^k$ for every $i \in [n]$), \mathbf{b}_i will implicitly denote the random linear combination of the elements, i.e. $\sum_{j=1}^k Y^{j-1} \mathbf{b}_{(i,j)}$, whenever it appears in a formula. For example,

$$\frac{1}{X + \mathbf{b}_i} = \frac{1}{X + \sum_{j=1}^k Y^{j-1} \mathbf{b}_{(i,j)}}.$$

This is a k -special-sound transformation, so a previously (a_1, \dots, a_μ) -special-sound protocol becomes (k, a_1, \dots, a_μ) -special sound after it.

⁸ Sorting takes $O(\ell \log \ell)$ time, but this is entirely prior to and not a part of our memory-proving protocol.

Achieving Perfect Completeness. The three protocols we introduce will not yet have perfect completeness since the prover will be sending over vectors of fractions of the form $\mathbf{h}_j = \frac{\mathbf{n}_j}{d_j} \quad \forall j \in [|\mathbf{h}|]$, where the computation of the denominator d is dependent on values in the given witness or lookup table. If there exists any value in some entry of the witness or lookup table such that $d = 0$, then the prover message will be undefined. We can achieve perfect completeness by following the same strategy for achieving perfect completeness in Π_{LK} in [BC23], which is to have the verifier set $\mathbf{h}_j = 0$ in this case and changing the verification equation from $\mathbf{h}_j \cdot \mathbf{d}_j \stackrel{?}{=} \mathbf{n}_j$ to

$$\mathbf{d}_j \cdot (\mathbf{h}_j \cdot \mathbf{d}_j - \mathbf{n}_j) \stackrel{?}{=} 0$$

The new check ensures that either $\mathbf{h}_j = \frac{\mathbf{n}_j}{d_j}$ or $\mathbf{d}_j = 0$. This increases the verifier degree in all of the three subprotocols to 3. Without these checks, the protocol has a negligible completeness error of $(\frac{\sum_i |\mathbf{h}_i|}{|\mathbb{F}|})$, where $\mathbf{h}_1, \mathbf{h}_2, \dots$ are the vectors of fractions sent by the prover. This completeness error is negligible. However, IVC and thus accumulation from which IVC is constructed require the protocols to be perfectly complete [BCLMS21] because IVC is designed for distributed computations where the continuance of computation is important, even on adversarially generated inputs.

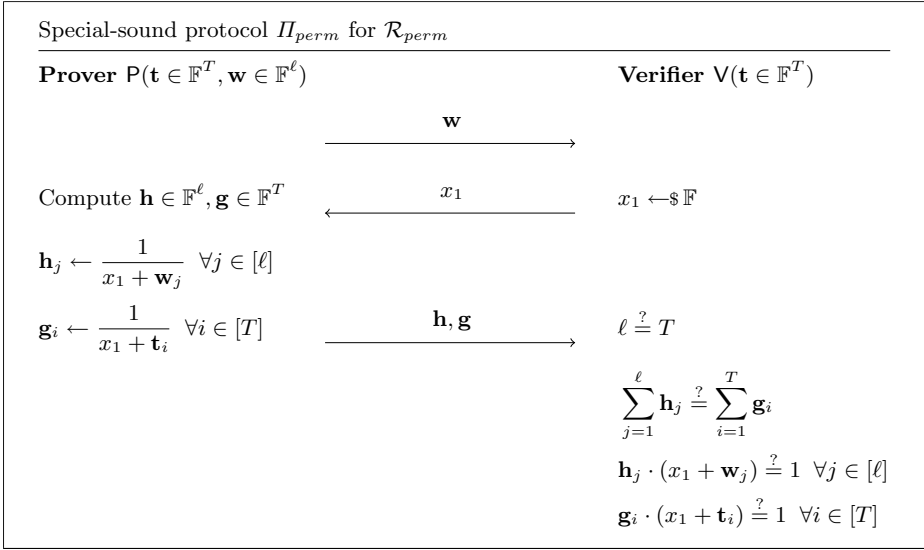
4.1 Checking Permutation Using Lookup Relation

Definition 9. (Definition 10 from [BC23]) Two sequences of field elements $\mathbf{w} = [\mathbf{w}_i]_{i=1}^n, \mathbf{t} = [\mathbf{t}_i]_{i=1}^n$ are in $\mathcal{R}_{\text{perm}}$ if there exists permutation $\sigma : [n] \rightarrow [n]$ such that for all $i \in [n], \mathbf{w}_i = \mathbf{t}_{\sigma(i)}$.

Lemma 3. Let \mathbb{F} be a field of characteristic $p > \max(\ell, T)$. Given two sequences of field elements $\mathbf{w} = [\mathbf{w}_i]_{i=1}^\ell$ and $\mathbf{t} = [\mathbf{t}_i]_{i=1}^T$, we have \mathbf{w}, \mathbf{t} are permutations of each other (i.e. \mathbf{w}, \mathbf{t} are in $\mathcal{R}_{\text{perm}}$) if and only if $\ell = T$ and

$$\sum_{i=1}^\ell \frac{1}{X + \mathbf{w}_i} = \sum_{i=1}^T \frac{1}{X + \mathbf{t}_i}. \tag{4}$$

We can therefore describe a special-sound protocol Π_{perm} for $\mathcal{R}_{\text{perm}}$ by simply adding the check $\ell \stackrel{?}{=} T$ and removing the need to compute \mathbf{m} from Π_{LK} for \mathcal{R}_{LK} in [BC23].



Complexity. Π_{perm} is a 3-move protocol (i.e. $k = 2$); the degree of the verifier is 2; the number of non-zero elements in the prover message is at most $2\ell + T$.

Special-Soundness. Just as Π_{LK} from [BC23], the perfect complete version of Π_{perm} is $2(\ell + T)$ -special-sound, assuming each entry $\mathbf{w}_j, \mathbf{t}_i$ is a single value for all $j \in [\ell], i \in [T]$.

4.2 Indexed-Vector Lookup Relation

Definition 10. (*Indexed-Vector Lookup Relation*) Given configuration $\mathcal{C}_{ivlk} := (T, \ell, \mathbf{t})$ where ℓ is the number of lookups and $\mathbf{t} \in \mathbb{F}^T$ is the lookup table, the triple $(\mathbf{t}, \mathbf{w} \in \mathbb{F}^\ell, \mathbf{b} \in \mathbb{F}^\ell)$ are in the relation \mathcal{R}_{ivlk} if for all $j \in [\ell], \mathbf{b}_j \in [T]$ and $\mathbf{w}_j = \mathbf{t}_{\mathbf{b}_j}$.

Lemmas 4 and 5 in the following are extensions on Lemma 4 and 5 from [Hab22], respectively.

Lemma 4. Let \mathbb{F} be an arbitrary field and $f_1, f_2 : \mathbb{F}^2 \rightarrow \mathbb{F}$ any functions. Then

$$\sum_{z_1, z_2 \in \mathbb{F}^2} \frac{f_1(z_1, z_2)}{X - z_1 \cdot Y - z_2} = \sum_{z_1, z_2 \in \mathbb{F}^2} \frac{f_2(z_1, z_2)}{X - z_1 \cdot Y - z_2} \tag{5}$$

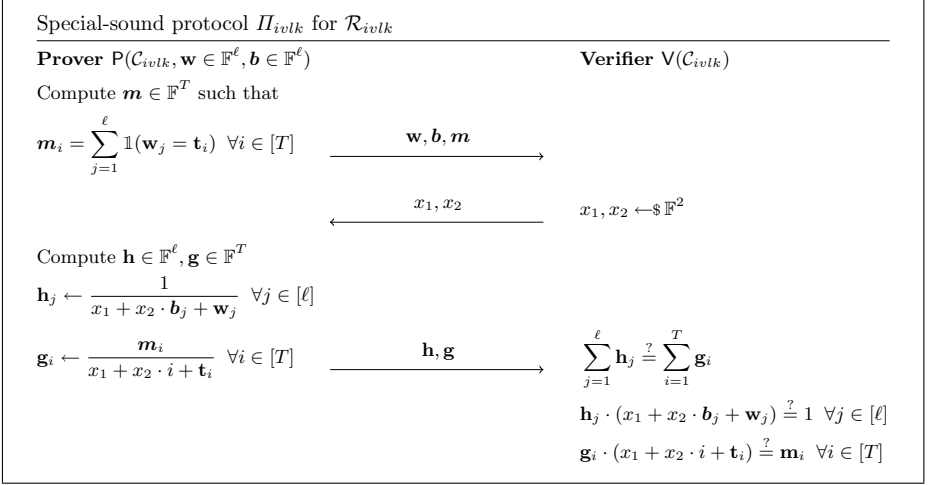
in the rational function field $\mathbb{F}(X, Y)$, if and only if $f_1(z_1, z_2) = f_2(z_1, z_2)$ for every $z_1, z_2 \in \mathbb{F}^2$.

Lemma 5. Let \mathbb{F} be a field of characteristic $p > \max\{\ell, T\}$. Given a sequence of field elements $\mathbf{w} \in \mathbb{F}^\ell, \mathbf{b} \in \mathbb{F}^\ell, \mathbf{t} \in \mathbb{F}^T$, we have $(T, \ell, \mathbf{t}, \mathbf{w}, \mathbf{b}) \in \mathcal{R}_{ivlk}$ if and only if the following equation holds in the function field $F(X, Y)$

$$\sum_{j=1}^{\ell} \frac{1}{X + Y \cdot \mathbf{b}_j + \mathbf{w}_j} = \sum_{i=1}^T \frac{\mathbf{m}_i}{X + Y \cdot i + \mathbf{t}_i} \tag{6}$$

where $\mathbf{m} = \{\mathbf{m}_i\}_{i=1}^T$ is the counter vector such that \mathbf{m}_i is the count of (i, \mathbf{t}_i) in (\mathbf{b}, \mathbf{w}) .

We can therefore describe a special-sound protocol for the indexed-vector lookup relation.



Complexity. Π_{ivlk} is a 3-move protocol (i.e. $k = 2$); the degree of the verifier is 3; the number of non-zero elements in the prover message is at most 5ℓ .

Lemma 6. Π_{ivlk} is $((\ell + T), 2(\ell + T))$ -special-sound.

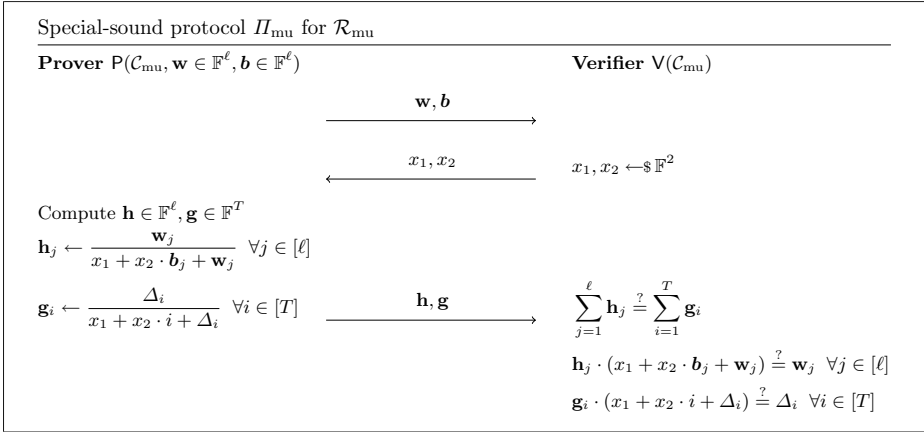
4.3 Mem-Update Relation

Definition 11 (Mem-Update Relation). Given configuration $\mathcal{C}_{mu} := (T, \ell, \Delta)$ where ℓ is the number of lookups and $\Delta \in \mathbb{F}^T$ is the update table, the triple $(\mathbf{w} \in \mathbb{F}^\ell, \mathbf{b} \in \mathbb{F}^\ell, \Delta)$ are in the relation \mathcal{R}_{mu} if for all $j \in [\ell]$, if $\mathbf{w}_j \neq 0$ then $\mathbf{w}_j = \Delta_{\mathbf{b}_j}$, and for all $i \in [T]$, if $\Delta_i \neq 0$ then there exists $j \in [\ell]$ such that $\mathbf{b}_j = i$ and $\Delta_i = \mathbf{w}_j$.

Lemma 7. Let \mathbb{F} be a field of characteristic $p > \max\{\ell, T\}$. Given the sequences of field elements $\mathbf{w} \in \mathbb{F}^\ell, \mathbf{b} \in \mathbb{F}^\ell, \Delta \in \mathbb{F}^T$, we have $(T, \ell, \Delta, \mathbf{w}, \mathbf{b}) \in \mathcal{R}_{mu}$ if and only if the following equation holds in the function field $F(X, Y)$

$$\sum_{j=1}^{\ell} \frac{\mathbf{w}_j}{X + Y \cdot \mathbf{b}_j + \mathbf{w}_j} = \sum_{i=1}^T \frac{\Delta_i}{X + Y \cdot i + \Delta_i} \quad (7)$$

We describe a $((\ell + T), 2(\ell + T))$ -special-sound protocol for the mem-update relation.



Complexity. Π_{mu} is a 3-move protocol (i.e. $k = 2$); the degree of the verifier is 3; the number of non-zero elements in the prover message is at most 4ℓ .

Lemma 8. Π_{mu} is $((\ell+T), 2(\ell+T))$ -special-sound, assuming each entry \mathbf{w}_j, Δ_i for all $j \in [\ell], i \in [T]$ is a single value.

Efficiency in Accumulation. We refer to Table 3 for an overview over the efficiency of the protocol. Importantly the prover time is entirely independent of T . The protocol can also be combined with our GKR protocol as layed out in Sect. 7. This reduces the prover time by eliminating the multi-scalar multiplication with full field elements. It is, thus, a useful option when the size of the table elements is significantly smaller than the field, e.g. 32-bit elements vs a 256-bit field.

Table 3. Efficiency Table for Accumulating Π_{mu} . We only list the dominant efficiency factors, ignoring the cost for \mathcal{P}_{acc} to compute the vectors. Column 2 refers to the total size of the prover messages. Here \mathbb{T} is the set of small elements that are stored in the table, whereas \mathbb{F} refers to full field elements. Column 3 is the verifier degree. Column 5 is the number of prover messages. Note that the number of messages in the GKR case can be further reduced with the optimizations mention in Sect. 7. Column 6 is the dominant factor in the prover time. An (a, B) -MSM refers to a multi-scalar multiplication of a scalars that are each within the set B . The MSM scales roughly linear in $|\log B|$. Column 7 is the number of group scalar multiplications the accumulation verifier performs.

	P Time	P Msg	deg(V)	# P Msgs	\mathcal{P}_{acc} Time	\mathcal{V}_{acc} Time
Plain	$O(\ell)$	$2\ell \mathbb{F} + 2\ell \mathbb{T}$	3	2	$(2\ell, \mathbb{T})$ -MSM + $(2\ell, \mathbb{F})$ -MSM	4G
With GKR	$O(\ell)$	$2\ell \mathbb{F} + 2\ell \mathbb{T}$	7	$(c + 1) \log T$	$(2\ell, \mathbb{T})$ -MSM + $O(\ell \log \ell) \mathbb{F}$	$(c + 1) \log T \mathbb{G}$

5 The LogUp-Powered Memory-Proving Algorithm

5.1 Using LogUp-Style Relations for Memory-Proving

See Fig. 1 for the full memory-proving protocol Π_{MP} .

Given the old memory $\text{OM} = [\mathbf{v}_i]_{i=1}^T$; $\text{Rd} = \text{RL} \parallel W = [(\mathbf{a}_i, \mathbf{v}_i, \mathbf{t}_i)]_{i=1}^k$ and $\text{Wr} = \text{WL} \parallel R = [(\mathbf{a}_i, \mathbf{v}_i, \mathbf{t}_i)]_{i=1}^k$, which were constructed as described in Sect. 3. Let $\ell := |W| = |R|$.

In Π_{MP} , the prover takes as input $(\text{OM}, \text{Rd} = \text{RL} \parallel W, \text{Wr} = \text{WL} \parallel R)$, and the verifier takes as input $(\text{OM}^V, \text{RL}, \text{WL})$, where OM^V is the verifier's stored state of the memory. At the start of the protocol, the prover sends R, W to the verifier, and the verifier checks that they are sorted in the same order by addresses, i.e. $R.\mathbf{a} \stackrel{?}{=} W.\mathbf{a}$. The rest of the protocol is composed of the following three LogUp-style protocols:

1. Use Π_{perm} to show that $(k, \text{Rd}, \text{Wr})$ are in $\mathcal{R}_{\text{perm}}$.
2. Use Π_{ivlk} to show that $(T, \ell, \text{OM}, \mathbf{r}, \mathbf{b})$ are in $\mathcal{R}_{\text{ivlk}}$, where $\mathbf{b} := R.\mathbf{a}$ and $\mathbf{r} := R.\mathbf{v}$.
3. Suppose W, R are all ordered by the addresses of the entries. The prover computes $\mathbf{w} := W.\mathbf{v} - R.\mathbf{v} \in \mathbb{F}^\ell$, $\mathbf{b} = R.\mathbf{a} \in \mathbb{F}^\ell$, and then use them to efficiently compute $\Delta \in \mathbb{F}^T$ as follows.

$$\forall i \in [T], \Delta_i = \begin{cases} \mathbf{w}_j & \text{if } i = \mathbf{b}_j \exists j \in [\ell] \\ 0 & \text{otherwise} \end{cases}$$

which the prover then use to efficiently compute the updated memory NM as follows:

$$\forall i \in [T], \text{NM}_i = \begin{cases} \text{OM}_i + \Delta_i & \text{if } i = \mathbf{b}_j \exists j \in [\ell] \\ \text{OM}_i & \text{otherwise} \end{cases}$$

This update only takes time linear in ℓ and independent of the total memory size T .

The prover then sends the NM to the verifier, who will compute \mathbf{w}, \mathbf{b} from R, W and Δ from NM, OM^V by himself, and they run Π_{mu} to show that $(T, \ell, \Delta, \mathbf{w}, \mathbf{b})$ are in \mathcal{R}_{mu} .

If all the check passes, the verifier accepts NM as the correctly updated memory. In the next round, the previously computed NM becomes the new OM, OM^V for the prover and the verifier, respectively.

Complexity. It is a 3-move protocol (i.e. $k = 2$); the degree of the verifier is 4; the number of non-zero elements in the prover message is at most $8k + 6\ell$. This is important because the prover pays linearly in the number of non-zero elements when computing the commitments. It is important to note that the total time of running the protocol is independent of T : running Π_{perm} is linear in k , and Π_{ivlk} and Π_{mu} are linear in ℓ ; the final step of computing the updated memory

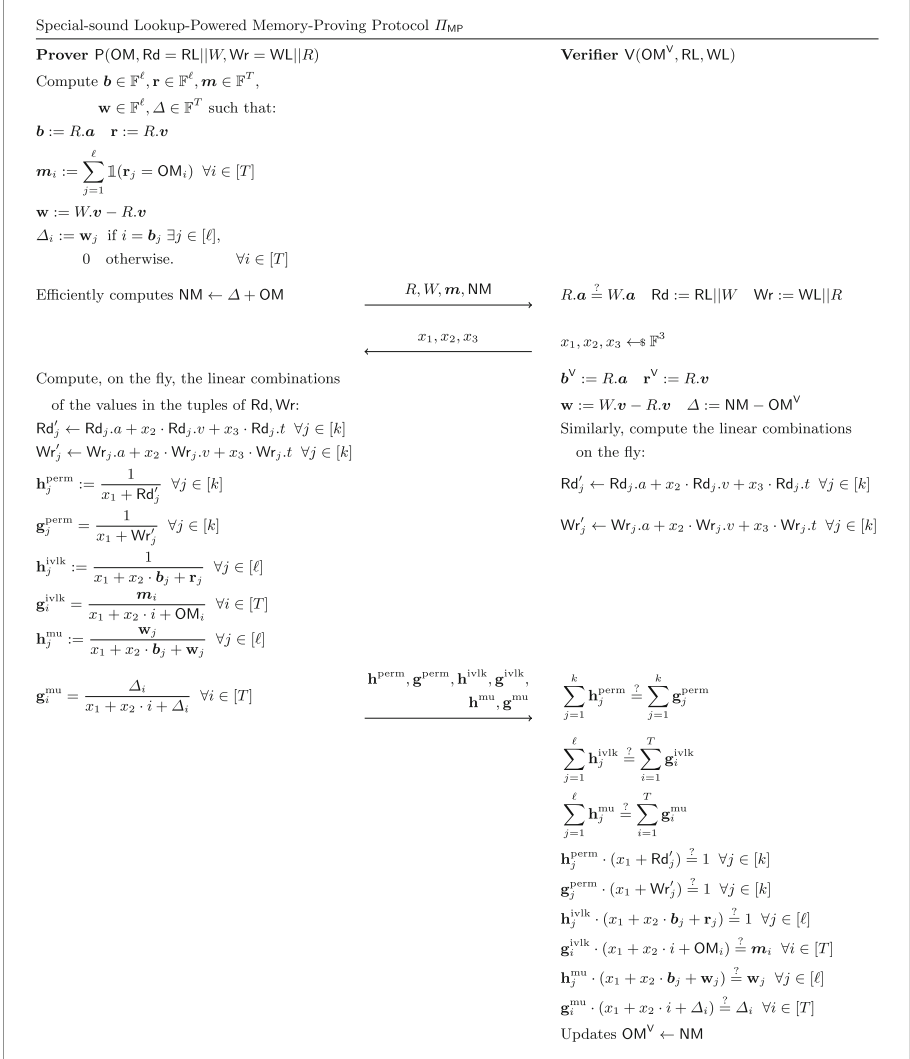


Fig. 1. Full LogUp-Powered Memory-Proving Algorithm Π_{MP}

can also be done in $O(\ell)$ time. As we assume $k \ll T$, i.e. the total number of entries in Rd, Wr are much smaller than the total size of the memory, the time it costs to run this memory-proving algorithm is $O(\ell)$ and independent of T .

Security. In this algorithm, Π_{perm} is $(3, 4k)$ -special-sound, Π_{ivlk} is $((\ell + T), 2(\ell + T))$ -special-sound, and Π_{mu} is $((\ell + T), 2(\ell + T))$ -special-sound. Therefore, the algorithm is $((\ell + T), 2(\ell + T))$ -special-sound overall.

Computing Commitments in the Accumulation Scheme. When we use the ProtoStar compiler to turn our memory-proving protocol Π_{MP} into an accumulation scheme, the resulting accumulation prover P_{acc} will send the homomorphic commitments to the prover messages instead of the plain vectors. The homomorphic commitments to the $O(\ell)$ -sized and ℓ -sparse vectors can all be computed in time independent of T since committing to 0 is free. Moreover, the commitment to NM can be computed in one step by adding the commitment to Δ and the commitment to OM .

Speeding up Memory-Proving with LogUp-GKR (described in Sect. 7). In the memory-proving protocol the prover’s messages are either $O(\ell)$ sized or $O(\ell)$ sparse. However, a more fine-grained view looks at the actual bit-length of the messages. When compiling to an IVC, the prover needs to commit to all the messages and this operation is linear in the bit-length of messages. In the first round of the protocol the prover sends R, W, \mathbf{m}, Δ . These values are representations of values read or written to memory, or their addresses and timestamps respectively. If the memory architecture only supports λ' -bit values, e.g. $\lambda' = 32$, then these values are all much smaller than the size of the field (which is proportional to the security parameter). In the second prover message, the prover sends multiple inverses. These values are large, even if the denominator itself is small. Note that all vectors are either $O(\ell)$ -sized or $O(\ell)$ -sparse.

Instead of sending the second round values and having the verifier perform the sum over the fractions, we will take the approach of LogUp [PH23], where the sum of fractions is computed using formal fractions. Importantly, this does not require sending the fractions itself. This can significantly reduce the prover cost as it now does not need to commit to λ -bit “full” field elements.

The bivariate GKR protocol for LogUp as described in Sect. 7, requires the prover to commit to messages of size $c \cdot T^{1/c}$ for any parameter c . We can set c such that $T^{1/c}$ is a marginal cost, compared to committing to the “small” numerators and denominators.

In Π_{MP} , some of the vectors of fractions sent by the prover are sparse (E.g. $\mathbf{g}^{\text{ivlk}}, \mathbf{g}^{\text{mu}}$). Even though they contain T entries in total, at most ℓ of them are non-zero. We can take advantage of this sparseness in LogUp GKR by setting $d_i = 1$ whenever $n_i = 0$ for all $i \in [T]$, and the prover will store $d_i - 1 = 0$ in its head to facilitate computation. [CMT12] shows that sumcheck is linear in the sparseness of the vector, which implies that GKR is also linear in the sparseness. Therefore, the time it takes to run LogUp-GKR for those sparse polynomials will be independent of its size.

It is not necessary to run LogUp-GKR from the sum over the entire vector. We can break the overall summation into a sum of several smaller summations, and run LogUp-GKR for each. This reduces the rounds of GKR, and we can then check the final sum in a straightforward manner.

After running GKR, we check that the two fractions are equal by checking the products of one numerator and the other denominator are equal.

Extending to Key-Value Store. Our protocol can be extended to prove the correctness of key-value store, which is very similar to memory access but the storage does not have a fixed size T . We describe the details of this extension in Appendix B.1 of the full version [BC24].

5.2 Accumulation Prover Runs in Time Independent of T

When we use the ProtoStar compiler to turn Π_{MP} into an accumulation scheme, the resulting P_{acc} will run in time independent of the memory size T , because the messages of the underlying special-sound prover, the cross error terms, and the updated accumulator can all be computed time independent of T . The details of these subalgorithms can be found in Appendix C of the full version [BC24].

Underlying Special-Sound Prover Runs in $O(\ell)$ Time. All computations of the prover in Π_{MP} can be done in $O(\ell)$ time. Vectors $\mathbf{h}^{\text{perm}}, \mathbf{g}^{\text{perm}}, \mathbf{h}^{\text{ivlk}}, \mathbf{h}^{\text{mu}}$ all have $O(\ell)$ size, so they can clearly be computed in $O(\ell)$ time. Vectors $\mathbf{m}, \Delta, \mathbf{g}^{\text{ivlk}}, \mathbf{g}^{\text{mu}}$ have size T , but they all have at most ℓ nonzero entries, so an honest prover only needs $O(\ell)$ time to compute them. Updating the memory also takes $O(\ell)$ time for an honest prover, since only Δ is sparse and only ℓ locations in the memory table need to be changed.

Computing the Cross Error Terms in $O(\ell)$ Time. In the following, we use acc to represent the accumulator, π the current proof, and acc' to represent the updated accumulator. We refer the readers to Sect. 3.4 in [BC23] for a general formula on how cross error terms $[\mathbf{e}_j]_{j=1}^{d-1}$ are computed in the accumulation scheme. P_{acc} will linearly combine the old accumulator and the current proof using a random challenge and use them as inputs to the decider (which is algebraic of degree d). For an honest prover, the zero coefficient of the polynomial should be the old accumulator's error term, and the highest-degree coefficient should be 0. The prover needs to then compute and commit to each of the coefficients in between (a.k.a. cross error terms). For most V_{sps} checks, it is intuitive how the cross error terms can be computed in $O(\ell)$ time, as the vectors will be either $O(\ell)$ -sized or ℓ -sparse. The detailed algorithm for computing the cross error term of the less intuitive $\mathbf{g}_i^{\text{ivlk}} \cdot (x_1 + x_2 \cdot i + \text{OM}_i) \stackrel{?}{=} \mathbf{m}_i \quad \forall i \in [T]$ check in time independent of T in the k th round of accumulation is given Appendix C.1 of the full version [BC24].

Note that this helper algorithm is only required when LogUp-GKR (described in Sect. 7) is *not* used. Using LogUp-GKR the cross error term computation (using the algorithms described in [BC23]) takes only $O(c \cdot T^{1/c}) = o(T)$ time, i.e. is insignificant compared to the rest of the prover computation.

Updating the Accumulator in $O(\ell)$ Time The prover still needs to compute the new accumulator $\text{acc}' \cdot \mathbf{g} \leftarrow \text{acc} \cdot \mathbf{g} + X \cdot \pi \cdot \mathbf{g}$ and $\text{acc}' \cdot \text{OM} \leftarrow \text{acc} \cdot \text{OM} + X \cdot \pi \cdot \text{OM}$. While computing $\text{acc}' \cdot \mathbf{g}$ clearly takes $O(\ell)$ time because $\pi \cdot \mathbf{g}$ is ℓ -sparse, the complexity for naively computing $\text{acc}' \cdot \text{OM}$ is linear in T . We show a trick in Appendix C.2 of the full version [BC24] that enables us to accumulate OM in time independent of T .

Overall Prover Efficiency. We display the efficiency metric of both the resulting plain protocol as well as the GKR-version in Table 4. The key prover efficiency is the P_{acc} Time. In the plain protocol, the prover first commits to R, W and \mathbf{m} . It also commits to Δ in order to compute the commitment to the updated memory NM . R, W are each of size ℓ and contain tuples of three elements (a, v, t) . Note that the a values will be exactly the same in R and W , so committing to R, W takes an MSM of size 5ℓ . Committing to Δ is an additional sparse MSM with ℓ non-zero elements. Committing to \mathbf{m} is a negligible cost as \mathbf{m} is a bit-vector. The prover also needs to commit to the vectors of fractions in the second round of the protocol. There are 6 such vectors that are either of size ℓ (for simplicity we assume $k = \ell$) or ℓ -sparse. Finally the accumulation prover needs to compute the cross terms for accumulation. We show how to do this in Sect. 5.2 and it requires an additional 3 ℓ -sparse MSMs. This results in a prover time that only requires committing to 15ℓ elements. We can replace the second round of the plain protocol using GKR. The GKR protocol requires committing to $O(T^{1/c})$ for an arbitrary constant c . This reduces the overall accumulation prover complexity to only 6ℓ elements, each of which is only as large as the elements stored in the table. Note that this is almost minimal, as even just recording a single read or write, already requires 3 elements, the address, the value and the timestamp.

Table 4. Efficiency Table for Accumulating Memory-Proving Protocol. See Table 3 for an explanation of the columns and symbols. For simplicity we assume that $k = \ell$. They are of the same order.

	P Time	P Msg	deg(V)	# P Msgs	P_{acc} Time	V_{acc} Time
Plain	$O(\ell)$	$5\ell T + 6\ell F$	3	2	$(6\ell, T)$ -MSM + $(9\ell, F)$ -MSM	$4G$
Using GKR	$O(\ell \log \ell)$	$5\ell T + O(T^{1/c})$	7	$(c + 1) \cdot \log T$	$(6\ell, T)$ -MSM	$(c + 1) \log T G$

6 Accumulation-Friendly GKR

Right now, the prover in our memory-proving IVC scheme needs to commit to 15 field elements per memory access, 9 of which are small memory entries, and 6 of which are large field elements, i.e. $\log |\mathbb{F}|$ -bit, since they are the inverses of the memory entries. As an example, say the memory only contains 32-bit entries. Using homomorphic commitments require fields of size at least 2^{256} , leading to a factor $256/32 = 8$ blowup when computing commitments.

An intuitive solution is to employ the GKR protocol, since it has the advantage of only requiring committing to the inputs/outputs and not any intermediate values of the circuit wires. Unfortunately, naively using GKR in accumulation results in $\log^2 n$ rounds (assuming n is the number of inputs), which is expensive since the ProtoStar accumulation compiler pays linearly in the number of rounds.

We design a version of the GKR protocol that is better suited for accumulation. It takes fewer rounds but retains the desired property of not requiring committing to any intermediate values. The core ingredient is a bivariate sumcheck protocol which only has two rounds.

6.1 Subprotocol for the Verifier to Efficiently Evaluate a Function

Bivariate sumcheck requires the verifier to evaluate polynomials of degree $\Theta(\sqrt{n})$, where n is the width of the GKR circuit. This is prohibitively large. Fortunately, we can transform evaluation into a low-degree check by sending additional witnesses. We describe the low-degree evaluation protocol Π_{eval} below.

Subprotocol Π_{eval} for evaluating $f : \mathbb{F}^k \rightarrow \mathbb{F}$ at some $\mathbf{a} \in (\mathbb{F} \setminus \mathbb{H})^k$ s.t. $m := |\mathbb{H}| > \deg(f)$

<p>Prover $P(f, \mathbf{a} = [a_1, \dots, a_k])$</p> <p>$\mathbf{a}^i \leftarrow [a_1^i, \dots, a_k^i] \quad \forall i \in \{2, 4, \dots, m\}$</p> <p>$A := (\mathbf{a}, \mathbf{a}^2, \mathbf{a}^4, \dots, \mathbf{a}^m)$</p> <p>$L_x^{\mathbb{H}}(u) := \frac{c_x(u^m - 1)}{u - x} \quad \forall x \in \mathbb{H}$</p>	$\xrightarrow{A, L_x^{\mathbb{H}}(u) \quad \forall x \in \mathbb{H}}$	<p>Verifier $V(f, \mathbf{a}, [f(\mathbf{x})]_{\mathbf{x} \in \mathbb{H}^k})$</p> <p>$A(0) \stackrel{?}{=} \mathbf{a} \quad A(i) \stackrel{?}{=} A(i-1)^2 \quad \forall i \in \{1, \dots, \log m - 1\}$</p> <p>$\prod_{j=1}^k L_{x_j}^{\mathbb{H}}(\mathbf{a}_j) \cdot \prod_{j=1}^k (\mathbf{a}_j - x_j) \stackrel{?}{=} \prod_{j=1}^k c_{x_j} \cdot (A(\log m, j) - 1) \quad \forall \mathbf{x} \in \mathbb{H}^k$</p> <p>$f(\mathbf{a}) \leftarrow \sum_{\mathbf{x} \in \mathbb{H}^k} \left(\prod_{j=1}^k L_{x_j}^{\mathbb{H}}(\mathbf{a}_j) f(\mathbf{x}) \right)$</p>
---	---	---

Efficiency. The verification degree is $2k$. The prover sends $m + k \cdot \log m$ values.

In the protocol above, \mathbb{H} is a multiplicative subgroup of \mathbb{F} , and we assume $m := |\mathbb{H}|$ is a multiple of 2. This implies that the i th element of \mathbb{H} is the i th root of unity and also that the Lagrange polynomial L_x has the form described above, where c_x is the barycentric weight. Note that P sends over a $\log m \times k$ matrix A . $A(i) := \mathbf{a}^{2^i}$ denotes the i th row of A , and $A(i, j) := \mathbf{a}_j^{2^i}$.

Security. The protocol has perfect completeness and soundness. The first line of checks ensure that the matrix A was computed correctly as claimed by the prover. In the second line of check, note that $A(\log m, j) = \mathbf{a}_j^{2^{\log m}} = \mathbf{a}_j^m$. Hence if the equality holds, we have

$$eq(\mathbf{x}, \mathbf{a}) = \prod_{j=1}^k \frac{c_{x_j}(\mathbf{a}_j^m - 1)}{\mathbf{a}_j - \omega_{x_j}} = \prod_{j=1}^k L_{x_j}^{\mathbb{H}}(\mathbf{a}_j) \quad \forall \mathbf{x} \in \mathbb{H}^k$$

which indicates that $eq(\mathbf{x}, \mathbf{a})$ was computed correctly as claimed by the verifier. This implies that the two polynomials $f(\mathbf{a})$ and $\sum_{\mathbf{x} \in \mathbb{H}^k} eq(\mathbf{x}, \mathbf{a})f(\mathbf{x})$ are equal on m^k points. Since both of these polynomials have degree strictly smaller than m , being equal on m^k points indicates that they are the same polynomial.

6.2 Bivariate Sumcheck

We describe a bivariate sumcheck protocol because the ProtoStar compiler pays linearly in the number of rounds, and hence the number of variables in sumcheck. While there is a tradeoff between the number of variables and the degree in each variable, high degrees can be tolerated in the final accumulation scheme because the decider only runs once.

Bivariate Sumcheck to prove $\sum_{x \in \mathbb{G}_1, y \in \mathbb{G}_2} f(x, y) = T$, where $\mathbb{G}_1, \mathbb{G}_2 \subset \mathbb{H}$ s.t. $m := \mathbb{H} = \deg(f) + 1$		
Prover $\mathsf{P}(f, T)$		Verifier $\mathsf{V}(f, T)$
$f_1(X) \leftarrow \sum_{y \in \mathbb{G}_2} f(X, y)$	$\xrightarrow{f_1(\omega_i) \ \forall i \in [m]}$	
	\xleftarrow{a}	$a \leftarrow \mathbb{F} \setminus \mathbb{H}$
$f_2(Y) \leftarrow f(a, Y)$	$\xrightarrow{f_2(\omega_i) \ \forall i \in [m]}$	
	\xleftarrow{b}	$b \leftarrow \mathbb{F} \setminus \mathbb{H}$
$T^* \leftarrow f_2(b)$	$\xrightarrow{T^*}$	Use Π_{eval} to evaluate $f_1(a), f_2(b)$
		$\sum_{x \in \mathbb{G}_1} f_1(x) \stackrel{?}{=} T$ $\sum_{y \in \mathbb{G}_2} f_2(y) \stackrel{?}{=} f_1(a)$ $T^* \stackrel{?}{=} f_2(b)$ $T^* \stackrel{?}{=} f(a, b)$

Security. The protocol is clearly perfectly complete. It is (m, m) -special-sound. For a fixed challenge a_i , to show that $f_2(Y) = f(a_i, Y)$ requires the equality to hold for $\deg(f_2) + 1 = \deg_Y(f) + 1 \leq \deg(f) + 1 = m$ different challenges for Y , i.e. b_1, \dots, b_m . Then, since $f_2(Y) = f(a_i, Y)$, checking whether $\sum_{y \in \mathbb{G}_2} f_2(y) = f_1(a_i)$ is equivalent to checking $\sum_{y \in \mathbb{G}_2} f(a_i, y) = f_1(a_i)$ for any fixed a_i . To show that $f_1(X) = \sum_{y \in \mathbb{G}_2} f(X, y)$ requires the equality to hold for $\deg(f_1) + 1 = \deg_X(f) + 1 \leq \deg(f) + 1 = m$ different challenges for X , i.e. a_1, \dots, a_m . Therefore, with m different challenges on X and m different challenges on Y , the verifier can be sure that $\sum_{x \in \mathbb{G}_1} f_1(x) = \sum_{x \in \mathbb{G}_1, y \in \mathbb{G}_2} f(x, y)$. Finally, since $\sum_{x \in \mathbb{G}_1} f(x) = T$, it is verified that $\sum_{x \in \mathbb{G}_1, y \in \mathbb{G}_2} f(x, y) = T$.

The number of P messages shown in Table 5 is the number when the polynomial f in the sumcheck is non-sparse. Since the polynomial f will be sparse (independent of the memory size T) when performing memory-proving using our LogUp-powered protocol, the actual number of P messages will be much smaller.

6.3 Batching Subprotocol for GKR

Table 5. Efficiency Table for Accumulating Bivariate SumCheck Using Subprotocol Π_{eval} ($n := |f| \geq m^2$). In most applications f will be a composition of multiple polynomials; in order to compute $f_1(X)$, the prover will need to perform FFTs which take $n \log n$ operations in \mathbb{F} .

P Time	P Msg	deg(V)	# P Msgs
$n \log n \mathbb{F}$	$4\sqrt{n} + o(\sqrt{n})\mathbb{F}$ or hashes	2	3

Description of the Batching Subprotocol for batching k sumchecks into one:

- Given a list of tuples $[(g_j \in \mathbb{F}[X_1, \dots, X_c], T_j \in \mathbb{F})]_{j=1}^k$ and \mathbb{H}^c , such that $\sum_{\mathbf{x} \in \mathbb{H}^c} g_j(\mathbf{x}) = T_j$ for all $j \in [k]$.
- V chooses $r \leftarrow_{\$} \mathbb{F}$ at random and sends it to P .
- V batches all k sumchecks checks into one as follows

$$\sum_{\mathbf{x} \in \mathbb{H}^c} f(\mathbf{x}) \stackrel{?}{=} \sum_{j=1}^k r^{j-1} T_j$$

for $f(\mathbf{x}) := \sum_{j=1}^k r^{j-1} g_j(\mathbf{x})$. Note that if $g_j(\mathbf{x}) = eq(\mathbf{z}_j, \mathbf{x})g(\mathbf{x})$ then $f(\mathbf{x}) = g(\mathbf{x}) \cdot (\sum_{j \in [k]} r^{j-1} \cdot eq(\mathbf{z}_j, \mathbf{x}))$

Efficiency. In GKR we call this protocol with $g_j(\mathbf{x}) = g(\mathbf{x}) \cdot eq(\mathbf{z}_j, \mathbf{x})$. This means that the complexity of the batched sumcheck is equivalent to the complexity of sumcheck over g plus evaluating a random linear combination of the eq functions. This is only a small additive overhead over a single sumcheck of g .

Security. The batching subprotocol is perfectly complete. It is k -special-sound. We can define the following degree $(k - 1)$ polynomial:

$$\begin{aligned} g(r) &:= \left(\sum_{\mathbf{x} \in \mathbb{H}^c} f(\mathbf{x}) \right) - \left(\sum_{j=1}^k r^{j-1} T_j \right) \\ &= \sum_{\mathbf{x} \in \mathbb{H}^c} \left(\sum_{j=1}^k r^{j-1} g_j(\mathbf{x}) \right) - \left(\sum_{j=1}^k r^{j-1} T_j \right) = \sum_{j=1}^k r^{j-1} \left(\sum_{\mathbf{x} \in \mathbb{H}^c} g_j(\mathbf{x}) - T_j \right) \end{aligned}$$

If $g(r)$ is the zero polynomial, then $\sum_{\mathbf{x} \in \mathbb{H}^c} f(\mathbf{x}) = \sum_{j=1}^k r^{j-1} T_j$. In order to get $g = 0$, we need $\deg(g) + 1 = k$ points of r at which $g(r) = 0$.

7 LogUp GKR Protocol Using the Batching Subprotocol

We incorporate the subprotocols described in Sect. 6 with LogUp-GKR [PH23], where the circuit is designed for computing the cumulative sums of the fractions

using projective coordinates for the additive group of \mathbb{F} . The full protocol is in Appendix D of the full version [BC24]. We will use this protocol to perform the verifier checks for the LogUp-style arguments in Π_{MP} .

Further Reducing Communication and Rounds. The bivariate GKR protocol only uses $3 \cdot \log^2(k)$ rounds and has communication complexity \sqrt{k} . This is significantly fewer rounds than GKR with the standard multi-linear sumcheck which would use $O(\log^2 k)$ rounds. In most cases the additional communication of \sqrt{k} is only marginal, as the prover needed to commit to the input and output layers (of size k). However, in particular when using the protocol with sparse inputs the \sqrt{k} may indeed become dominant.

c -Variate Sumcheck. Fortunately, we can naturally generalize the protocol by relying on a c -variate sumcheck. In this case, the protocol has $(c + 1) \cdot \log_2(k)$ rounds but the communication complexity is only $O(c \cdot k^{1/c})$. This exponentially decays as c gets bigger. In the protocol we would expand the dimension in each variable, one by one, such that the size of the layer still grows by a factor of 2 in each round.

Higher Degree Reductions. Another optimization is to combine 2 rounds of GKR into one. This increases the degree of the GKR round polynomial by a factor of 2 but also decreases the number of rounds by the same factor. Using the ProtoStar compiler we only pay for the highest degree verification check, so this optimizations is particularly useful if the circuit already contains high degree checks.

Splitting the Summation for Round Reduction. The core motivation for proving the fractional sum within GKR instead of proving it directly, is that the prover does not need to commit to the inverses. When the numerator and denominator are composed of c -bit values and $\log |\mathbb{F}| = \Theta(\lambda)$ then this can reduce the commitment cost from $O(\lambda m)$ to just $O(c \cdot m)$, i.e. save a factor of $\frac{\lambda}{c}$. Note that the circuit computed by GKR has a triangle form and each layer is half the width of its parent layer. We can take advantage of this by splitting the sum into p parts each of $\frac{m}{p}$, component. The prover would need to commit to the outputs of each sum, i.e. p fractions. The total commitment cost is $O(c \cdot m + \lambda p)$. As long as $p \geq \frac{c \cdot m}{\lambda}$, the total commitment cost is still $O(c \cdot m)$. However, the sums computed within GKR are now significantly smaller, and only $\log \lambda - \log c$ GKR layers are required. A similar optimization applies when the input layer is sparse; however, then more layers are required to significantly bring down the cost of committing to the dense output layer.

7.1 Other Applications of GKR in IVC

GKR has many applications beyond the use in lookup protocols. For instance, GKR can be used to more efficiently prove that a scalar multiplication was

Table 6. Efficiency Table for Accumulating GKR. See Table 3 for an explanation of the columns. Here, n is width of the GKR circuit, c is the number of variant in the sumcheck protocol, and k is the degree of the sumcheck polynomial.

Variant	P Time	P Msg	deg(V)	# P Msgs	P_{acc} Time	V_{acc} Time
bivariate Σ	$O(n \log n)$	$O(n^{1/2})$	7	$3 \log n$	$O(n^{1/2})$ -MSM $+O(n \log n)\mathbb{F}$	$3 \log n + 2G$
c -variate Σ	$O(n \log n)$	$O(c \cdot n^{1/c})$	7	$(c + 1) \log n$	$O(c \cdot n^{1/c})$ -MSM $+O(n \log n)\mathbb{F}$	$(c + 1) \log n$ $+2G$
k -round GKR	$O(n \log n)$	$O(c \cdot n^{1/c})$	7	$(c + 1)k$	$O(c \cdot n^{1/c})$ -MSM $+O(n \log n)\mathbb{F}$	$(c + 1) \cdot k$ $+2G$

done correctly. This is particularly intriguing as group scalar multiplications are the most expensive operations within the recursive circuit. Concretely the GKR circuit for group scalar multiplication takes as input, a scalar s in bit representation $s_{\lambda-1} \dots s_1 s_0$ where s_i is either 0 or 1 for every $i \in \{0, \dots, \lambda - 1\}$ and $s_{\lambda-1}$ is the most significant bit, a base elliptic curve point in projective coordinates (X, Y, Z) , and an output curve point also in projective coordinates. The reason to use projective coordinates is that the double-and-add operation can be represented using low-degree (specifically degree 11) algebraic formulas [RCB16]. Using GKR, the prover would only need to commit to 6 scalars and λ bits. However, the depth of the circuit might be a bottleneck. We can further reduce the number of layers by providing more intermediary values. E.g. by providing k additional curve points, we can reduce the depth from λ to $\lambda/(k+1)$.

Acknowledgments. We would like to thank Arasu Arun and Lev Soukhanov for inspiring conversations on memory-checking and accumulation for GKR. We thank Shang Gao for pointing out several typos throughout our paper. We would also like to thank Sebastian Angel for the discussion on Spice and key-value store. Finally, we are grateful to all the reviewers for their helpful feedback.

References

- [AFK22] Thomas Attema, Serge Fehr, and Michael Kloof. “Fiat-Shamir Transformation of Multi-round Interactive Proofs”. In: *TCC 2022, Part I*. Ed. by Eike Kiltz and Vinod Vaikuntanathan. Vol. 13747. LNCS. Springer, Heidelberg, Nov. 2022, pp. 113–142. DOI: https://doi.org/10.1007/978-3-031-22318-1_5.
- [APPK24] Kasra Abbaszadeh, Christodoulos Pappas, Dimitrios Papadopoulos, and Jonathan Katz. *Zero-Knowledge Proofs of Training for Deep Neural Networks*. Cryptology ePrint Archive, Paper 2024/162. <https://eprint.iacr.org/2024/162.2024>. URL: <https://eprint.iacr.org/2024/162>.
- [AST23] Arasu Arun, Srinath Setty, and Justin Thaler. *Jolt: SNARKs for Virtual Machines via Lookups*. Cryptology ePrint Archive, Paper 2023/1217. <https://eprint.iacr.org/2023/1217>. 2023. URL: <https://eprint.iacr.org/2023/1217>.

- [BC23] Benedikt Bünz and Binyi Chen. *ProtoStar: Generic Efficient Accumulation/Folding for Special Sound Protocols*. Cryptology ePrint Archive, Paper 2023/620. <https://eprint.iacr.org/2023/620>. 2023. URL: <https://eprint.iacr.org/2023/620>.
- [BC24] Benedikt Bünz and Jessica Chen. *Proofs for Deep Thought: Accumulation for large memories and deterministic computations*. Cryptology ePrint Archive, Paper 2024/325. 2024. URL: <https://eprint.iacr.org/2024/325>.
- [BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. “Recursive composition and bootstrapping for SNARKS and proof-carrying data”. In: *45th ACM STOC*. Ed. by Dan Boneh, Tim Roughgarden, and Joan Feigenbaum. ACM Press, June 2013, pp. 111–120. DOI: <https://doi.org/10.1145/2488608.2488623>.
- [BCLMS21] Benedikt Bünz, Alessandro Chiesa, William Lin, Pratyush Mishra, and Nicholas Spooner. “Proof-Carrying Data Without Succinct Arguments”. In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 681–710. DOI: https://doi.org/10.1007/978-3-030-84242-0_24.
- [BCMS20] Benedikt Bünz, Alessandro Chiesa, Pratyush Mishra, and Nicholas Spooner. “Recursive Proof Composition from Accumulation Schemes”. In: *TCC 2020, Part II*. Ed. by Rafael Pass and Krzysztof Pietrzak. Vol. 12551. LNCS. Springer, Heidelberg, Nov. 2020, pp. 1–18. DOI: https://doi.org/10.1007/978-3-030-64378-2_1.
- [BCTV14] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. “Scalable Zero Knowledge via Cycles of Elliptic Curves”. In: *CRYPTO 2014, Part II*. Ed. by Juan A. Garay and Rosario Gennaro. Vol. 8617. LNCS. Springer, Heidelberg, Aug. 2014, pp. 276–294. DOI: https://doi.org/10.1007/978-3-662-44381-1_16.
- [BDFG21] Dan Boneh, Justin Drake, Ben Fisch, and Ariel Gabizon. “Halo Infinite: Proof-Carrying Data from Additive Polynomial Commitments”. In: *CRYPTO 2021, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. LNCS. Virtual Event: Springer, Heidelberg, Aug. 2021, pp. 649–680. DOI: https://doi.org/10.1007/978-3-030-84242-0_23.
- [BEGKN91] Manuel Blum, William S. Evans, Peter Gemmell, Sampath Kannan, and Moni Naor. “Checking the Correctness of Memories”. In: 32nd FOCS. IEEE Computer Society Press, Oct. 1991, pp. 90–99. DOI: <https://doi.org/10.1109/SFCS.1991.185352>.
- [BFRSBW13] Benjamin Braun, Ariel J. Feldman, Zuocheng Ren, Srinath Setty, Andrew J. Blumberg, and Michael Walfish. *Verifying Computations with State (Extended Version)*. Cryptology ePrint Archive, Report 2013/356. <https://eprint.iacr.org/2013/356>. 2013.
- [BGH19] Sean Bowe, Jack Grigg, and Daira Hopwood. *Halo: Recursive Proof Composition without a Trusted Setup*. Cryptology ePrint Archive, Report 2019/1021. <https://eprint.iacr.org/2019/1021>. 2019.
- [CBBZ22] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. *HyperPlonk: Plonk with Linear-Time Prover and High-Degree Custom Gates*. Cryptology ePrint Archive, Report 2022/1355. <https://eprint.iacr.org/2022/1355>. 2022.

- [CDvGS03] Dwaine E. Clarke, Srinivas Devadas, Marten van Dijk, Blaise Gassend, and G. Edward Suh. “Incremental Multiset Hash Functions and Their Application to Memory Integrity Checking”. In: *ASIACRYPT 2003*. Ed. by Chi-Sung Laih. Vol. 2894. LNCS. Springer, Heidelberg, 2003, pp. 188–207. DOI: https://doi.org/10.1007/978-3-540-40061-5_12.
- [CMT12] Graham Cormode, Michael Mitzenmacher, and Justin Thaler. “Practical verified computation with streaming interactive proofs”. In: *ITCS 2012*. Ed. by Shafi Goldwasser. ACM, Jan. 2012, pp. 90–112. DOI: <https://doi.org/10.1145/2090236.2090245>.
- [COS20] Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. “Fractal: Post-quantum and Transparent Recursive Proofs from Holography”. In: *EUROCRYPT 2020, Part I*. Ed. by Anne Canteaut and Yuval Ishai. Vol. 12105. LNCS. Springer, Heidelberg, May 2020, pp. 769–793. DOI: https://doi.org/10.1007/978-3-030-45721-1_27.
- [DNRV09] Cynthia Dwork, Moni Naor, Guy N. Rothblum, and Vinod Vaikuntanathan. “How Efficient Can Memory Checking Be?” In: *TCC 2009*. Ed. by Omer Reingold. Vol. 5444. LNCS. Springer, Heidelberg, Mar. 2009, pp. 503–520. DOI: https://doi.org/10.1007/978-3-642-00457-5_30.
- [EFG22] Liam Eagen, Dario Fiore, and Ariel Gabizon. *cq: Cached quotients for fast lookups*. Cryptology ePrint Archive, Report 2022/1763. <https://eprint.iacr.org/2022/1763>. 2022.
- [EG23] Liam Eagen and Ariel Gabizon. *ProtoGalaxy: Efficient ProtoStarstyle folding of multiple instances*. Cryptology ePrint Archive, Paper 2023/1106. <https://eprint.iacr.org/2023/1106>. 2023. URL: <https://eprint.iacr.org/2023/1106>.
- [GK22] Ariel Gabizon and Dmitry Khovratovich. *flookup: Fractional decompositionbased lookups in quasi-linear time independent of table size*. Cryptology ePrint Archive, Report 2022/1447. <https://eprint.iacr.org/2022/1447>. 2022.
- [Hab22] Ulrich Haböck. *Multivariate lookups based on logarithmic derivatives*. Cryptology ePrint Archive, Report 2022/1530. <https://eprint.iacr.org/2022/1530>. 2022.
- [KS23] Abhiram Kothapalli and Srinath Setty. *HyperNova: Recursive arguments for customizable constraint systems*. Cryptology ePrint Archive, Paper 2023/573. <https://eprint.iacr.org/2023/573>. 2023. URL: <https://eprint.iacr.org/2023/573>.
- [KST22] Abhiram Kothapalli, Srinath Setty, and Ioanna Tzialla. “Nova: Recursive Zero-Knowledge Arguments from Folding Schemes”. In: *CRYPTO 2022, Part IV*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13510. LNCS. Springer, Heidelberg, Aug. 2022, pp. 359–388. DOI: https://doi.org/10.1007/978-3-031-15985-5_13.
- [PH23] Shahar Papini and Ulrich Haböck. *Improving logarithmic derivative lookups using GKR*. Cryptology ePrint Archive, Paper 2023/1284. <https://eprint.iacr.org/2023/1284>. 2023. URL: <https://eprint.iacr.org/2023/1284>.
- [PK22] Jim Posen and Assimakis A. Kattis. *Caulk+: Table-independent lookup arguments*. Cryptology ePrint Archive, Report 2022/957. <https://eprint.iacr.org/2022/957>. 2022.

- [RCB16] Joost Renes, Craig Costello, and Lejla Batina. “Complete Addition Formulas for Prime Order Elliptic Curves”. In: *EUROCRYPT 2016, Part I*. Ed. by Marc Fischlin and Jean-Sébastien Coron. Vol. 9665. LNCS. Springer, Heidelberg, May 2016, pp. 403–428. DOI: https://doi.org/10.1007/978-3-662-49890-3_16.
- [SAGL18] Srinath Setty, Sebastian Angel, Trinabh Gupta, and Jonathan Lee. *Proving the correct execution of concurrent services in zeroknowledge*. Cryptology ePrint Archive, Report 2018/907. <https://eprint.iacr.org/2018/907>. 2018.
- [STW23] Srinath Setty, Justin Thaler, and Riad Wahby. *Unlocking the lookup singularity with Lasso*. Cryptology ePrint Archive, Paper 2023/1216. <https://eprint.iacr.org/2023/1216>. 2023. URL: <https://eprint.iacr.org/2023/1216>.
- [Val08] Paul Valiant. “Incrementally Verifiable Computation or Proofs of Knowledge Imply Time/Space Efficiency”. In: *TCC 2008*. Ed. by Ran Canetti. Vol. 4948. LNCS. Springer, Heidelberg, Mar. 2008, pp. 1–18. DOI: https://doi.org/10.1007/978-3-540-78524-8_1.
- [ZBKMNS22] Arantxa Zapico, Vitalik Buterin, Dmitry Khovratovich, Mary Maller, Anca Nitulescu, and Mark Simkin. “Caulk: Lookup Arguments in Sub-linear Time”. In: *ACM CCS 2022*. Ed. by Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi. ACM Press, Nov. 2022, pp. 3121–3134. DOI: <https://doi.org/10.1145/3548606.3560646>.
- [ZGKMR22] Arantxa Zapico, Ariel Gabizon, Dmitry Khovratovich, Mary Maller, and Carla Ràfols. *Baloo: Nearly Optimal Lookup Arguments*. Cryptology ePrint Archive, Report 2022/1565. <https://eprint.iacr.org/2022/1565>. 2022.



HELIOPOLIS: Verifiable Computation over Homomorphically Encrypted Data from Interactive Oracle Proofs is Practical

Diego F. Aranha¹(✉), Anamaria Costache², Antonio Guimaraes³,
and Eduardo Soria-Vazquez⁴

¹ Aarhus University, Aarhus, Denmark
dfaranha@cs.au.dk

² NTNU, Trondheim, Norway
anamaria.costache@ntnu.no

³ IMDEA Software Institute, Madrid, Spain
antonio.guimaraes@imdea.org

⁴ Technology Innovation Institute, Abu Dhabi, UAE
eduardo.soria-vazquez@tii.ae

Abstract. Homomorphic encryption (HE) enables computation on encrypted data, which in turn facilitates the outsourcing of computation on private data. However, HE offers no guarantee that the returned result was honestly computed by the cloud. In order to have such guarantee, it is necessary to add verifiable computation (VC) into the system.

The most efficient recent works in VC over HE focus on verifying operations on the ciphertext space of the HE scheme, which usually lacks the algebraic structure that would make it compatible with existing VC systems. For example, multiplication of ciphertexts in the current most efficient HE schemes requires non-algebraic operations such as real division and rounding. Therefore, existing works for VC over HE have to either give up on those efficient HE schemes, or incur a large overhead (an amount of constraints proportional to the ciphertext ring's size) in order to emulate these non-algebraic operations.

In this work, we move away from that paradigm by placing the verification checks in the *plaintext space* of HE, all while the prover remains computing on ciphertexts. We achieve this by introducing a general transformation for Interactive Oracle Proofs (IOPs) to work over HE, whose result we denote as HE-IOPs. We apply this same transformation to the FRI [Ben-Sasson et al., ICALP 2018] IOP of proximity and we show how to compile HE-Reed Solomon-encoded IOPs and HE- δ -correlated-IOPs with HE-FRI into HE-IOPs. Furthermore, our construction is compatible with a prover that provides input in zero-knowledge, and only relies on building blocks that are plausibly quantum-safe.

Aligning the security parameters of HE and FRI is a difficult task for which we introduce several optimizations. We demonstrate their efficiency with a proof-of-concept implementation and show that we can run FRI's commit phase for 4096 encrypted Reed Solomon codewords with degree bound 2^{11} in just 5.4 s (using 32 threads) on a `c6i.metal` instance

using less than 4GB of memory. Verification takes just 12.3 milliseconds (single-threaded) for the same parameter set and can be reduced to just 5.6ms with parameters optimized for the verifier.

1 Introduction

There are many usability and economic benefits for citizens and companies to outsource data storage/processing to remote servers, but cloud computing brings significant integrity and privacy risks. Homomorphic Encryption (HE) has been referred to by many as the “holy grail” technology to address the privacy risks of outsourced computing. Since the first scheme introduced by Gentry in 2009 [Gen09b], numerous advances and improvements have followed [BGV12, Bra12, FV12, CGGI20, CKKS17]. In particular, a privacy-preserving variant of the use-case of Machine Learning as a Service (MLaaS) has been shown to be particularly suitable for HE by a recent line of work [BMMP18, BCCW19, BGBE19].

However, HE by itself does not guarantee the integrity of the computing party. Dealing with this issue falls within the scope of Verifiable Computation (VC), which describes a collection of techniques ensuring that the output returned by the cloud servers is indeed the honest result of applying the requested function to the designated data. On the other hand, VC on its own does not protect the privacy of the outsourced yet sensitive data from the clients.

The cryptographic community realized that VC and HE are very complementary, since the limitations of the one are perfectly covered by the features of the other. Combining the two techniques is often referred to as “privacy-preserving verifiable computation” or “verifiable computation on encrypted data”. The first solution was proposed by Gennaro, Gentry and Parno [GGP10] in 2010, and employs a heavy combination of Yao’s garbled circuit for an one-time verifiable computation together with HE to reuse the garbled circuit across many inputs.

The later work of [FGP14] is efficient, but very limited in expressiveness. The use of homomorphic MACs limits the application to depth-1 circuits and requires to keep a secret verification key hidden from the prover, hence eliminating public verifiability. The work of [FNP20] improves the expressiveness of [FGP14], by allowing to efficiently compute circuits of (arbitrary) constant depth. Nevertheless, they only deal with a very narrow subset of inefficient HE schemes: a variant of BV [BV14], where the integer ciphertext modulus q matches the (prime) order of the source groups in the underlying pairing-based SNARK.

Both [GNS23, BCFK21] overcome the limitation in the selection of HE schemes in [FNP20] by supporting an arbitrary rather than a prime ciphertext modulus q . Still, for both works, dealing with the more complex HE operations such as modulo switching and key switching is very expensive. This makes it unclear whether it would be more practical to support efficient but complex schemes such as BGV and BFV, or BV with a potentially non-prime q .

The main problem for all those works is that they verify whether certain operations are done on (simplified versions of) the ciphertext space. Furthermore, they need to emulate the arithmetic of HE ciphertexts in one way or

another, incurring large overheads. While the addition of ciphertexts can be easily emulated (as addition of elements in the ciphertext space R_q^2), the product of ciphertexts is less algebraically structured and hard to emulate, since it mixes a number of computational steps such as bit-wise operations, real division, rounding, the product of elements in R_q and changing q during modulo switching operations.

As an example of how expensive these techniques were, consider trying to emulate the HE arithmetic within R_q (as in [GNS23, FNP20]), which is arguably the closest algebraic structure. Every bit-wise operation involved in the product of ciphertexts, such as rounding (present in the HE.ModSwitch operation in BGV and BFV), has the cost of one constraint¹ per bit of the ciphertext ring R_q .

This cost increases rapidly as the multiplicative depth of the circuit grows, not only because of the number of such operations but also because of how the HE parameters (including ciphertext size) grow accordingly. In the BGV scheme (as well as the BFV and CKKS one), increasing the multiplicative depth of the circuit by one usually requires to add a prime to the prime chain that makes up the ciphertext modulus. In more practical terms, this corresponds to increasing the ciphertext modulus by 30 – 50 bits every time we increase the multiplicative depth by one. This means that the ciphertext modulus will grow exponentially with the depth d of the circuit that one wants to evaluate (this takes the security level into account; see for example [APS15, CP19]). Alternatively, works like [BCFK21] circumvent the issue of doing bit-wise operations by using the BV scheme. In their work, the size of the ciphertext ring grows exponentially with the number of ciphertext-ciphertext multiplications that one wants to evaluate.

In this work, we deviate from this paradigm by enabling the verification of operations on the *plaintext space* of the HE scheme. At a high level, we show how to adapt holographic IOP-based SNARKs so that, on the one hand, the prover computes obliviously on the encrypted values while, on the other hand, the verifier performs the verification checks on the plaintext space. We choose to focus on holographic IOPs as a departure point for our verifiable computation protocol, since the holography property is particularly well suited for outsourcing scenarios. Nevertheless, our techniques could easily be adapted to non-holographic IOPs. We call our overall framework HEIOPolis, since its central components are homomorphic encryption (HE) and Interactive Oracle Proofs (IOPs).

1.1 Technical Overview

We manage to move verification from the ciphertext to the plaintext space by replacing the IOP oracles \mathcal{O} with *encrypted* oracles \mathcal{O}_{HE} , which are oracles to data that is homomorphically encrypted. While the prover \mathcal{P} does not know *what* the plaintexts they are computing on are, \mathcal{P} knows how to arrange them into oracles (i.e. \mathcal{P} can place $\text{HE.Enc}(x)$ into an oracle \mathcal{O}_{HE} , rather than x into \mathcal{O}). Whereas the modified IOP (denoted HE-IOP) can now only be verified by whoever has

¹ The number of *constraints* in R1CS or other models of computation are the main metric for the efficiency in SNARKs.

the HE decryption key², this new abstraction is very powerful: not only is the prover much more efficient, it is also very simple to reduce the security of an HE-IOP to that of its corresponding IOP (Theorem 2). Moreover, we also adapt several results in the literature compiling different variants of (zk)IOPs into (zk)SNARKs [COS20, BGK+23]. Our resulting zkSNARKs are plausibly post-quantum, since so are the BCS transform [BCS16, CMS19] and all the efficient HE schemes we have today.

Once oracles are replaced with *encrypted* oracles, our approach is black box on the different components of these compilers. A central part of these is the use of an IOP of proximity (IOPP) to Reed-Solomon Codes, which is interpreted either as a low degree test or a correlated agreement test [BCI+20, BGK+23]. As done in practice for unencrypted IOPs, we choose FRI [BBHR18] to instantiate this IOPP component. FRI is, a priori, particularly HE-friendly, in the sense that it only runs linear operations on the functions being tested, and products in HE are particularly expensive. Nevertheless, there are several challenges when trying to align the security parameters of HE schemes and FRI. This constitutes a significant part of our work, for which we discuss trade-offs and optimizations (Sect. 6) as well as we provide experimental data (Sect. 7).

Aligning Security Parameters. The first challenge is enabling FRI to work over a field of size $|\mathbb{F}_{p^D}| \approx 2^{256}$ for some prime p . This ensures that FRI remains secure when making it non-interactive through Fiat-Shamir for any Reed-Solomon codeword we would encounter in practice when compiling IOPs [BGK+23]. Using p^D as a plaintext modulus in the HE scheme would result in unmanageable parameters. We address this by emulating \mathbb{F}_{p^D} arithmetic using D ciphertexts, each encrypting elements from \mathbb{F}_p .

Reducing Depth and Exploiting HE Packing. A second challenge to the homomorphic evaluation of FRI is its multiplicative depth. Although it only involves multiplications between plaintexts and ciphertexts, the noise level can increase almost as much as with ciphertext multiplications, since plaintext are random elements of \mathbb{F}_p . A typical implementation of FRI would have depth $2n$ for an input of size 2^n , which represents a performance challenge for HE schemes. We solve this problem by introducing low-depth versions of every sub-routine required to evaluate FRI, and show how to perform Reed-Solomon codeword encoding with small fixed depth, as opposed to the traditional depth- n methods. We also propose a “Shallow Fold” algorithm to replace FRI’s standard `Fold` operation, which reduces the depth to 1 (from n), at the cost of increasing the complexity to $O(2^n \log(2^n))$ (from $O(2^n)$), which does not change the overall asymptotics. Additionally, we exploit packing within the HE scheme to further reduce the cost of Reed-Solomon encoding. In more detail, we consider packing methods that trade off memory consumption and execution time to accelerate the prover.

² In concurrent work on IOPs over encrypted data [GGW23], the authors discuss the use of fully homomorphic *commitments* [GVW15] as a way to recover public verifiability, but all known constructions for such primitive are very inefficient.

Minimizing HE Overhead for the Verifier. Finally, we take advantage of technique proposed in [CGGI20, CLOT21] to implement a repacking and recomposing technique, which significantly reduces the overhead of ciphertext decryption for the verifier. During the commit phase of FRI, the prover performs computations using RLWE samples of dimension N encrypting N messages in \mathbb{F}_p . During the query phase, however, the verifier only needs to learn two evaluation points in \mathbb{F}_{p^D} per round for each linearity check. If those ciphertexts are fully packed, an overhead of at least $N/(2D)$ is introduced. In order to avoid this, we extract the evaluation points from the RLWE samples of dimension N and repack them in another RLWE sample, but of a much smaller dimension, reducing decryption costs up to 128 times depending on the selected parameters. One key observation is that at this point, we *do not need to preserve any homomorphic properties*, as the verifier does not perform any further operations on these. Indeed, once the commit phase is finished, we can view the evaluation points as simply strings of bits, and our goal then becomes to encrypt them in the smallest possible ciphertext. This also makes the HE parameters adopted by the verifier completely independent of the ones adopted by the prover or of the input size 2^n .

All our optimizations are specifically targeted for FRI. Whether other existing IOPs of proximity (e.g. [ACY23]) or new ones could be better aligned in practice with HE schemes such as BGV and BFV is an interesting open work that our theoretical machinery already supports.

Proof-of-Concept Implementation. To demonstrate the practicality of our construction, we implement a proof-of-concept over the FRI implementation of Szepieniec *et al.* [S+21]. We extend it to work over non-prime fields and connect it to optimized FHE libraries. We test two parameter sets for encrypted code-words of size up to 2^{16} representing polynomials with degree bound up to $d = 2^{15}$. For a batch of 4096 polynomials with degree bound $d = 2^{11}$, our implementation takes just 5.4s to run FRI’s commit phase (including the Reed-Solomon code encoding) on 32 threads in a `c6i.metal` instance on AWS and requires less than 4GB of memory. Verification is much faster, taking just 12.3ms single-threaded (also for a batch of 4096 polynomials). With a parameter set optimized for the verifier, verification time drops to just 5.6ms single-threaded, at the cost of increasing the prover execution to 1.3 min. Our implementation is publicly available at <https://github.com/antoniocgj/HELIOPOLIS>.

Oracle Attacks. Moving verification from the ciphertext to the plaintext space inherently opens up for side-channel and composition attacks. If the verifier signals to the prover whether verification passes, this leaks 1 bit of information about the plaintext/secret key. A priori, it could seem that reusing the plaintext output obtained by the verifier as input to another protocol could also leak about the secret key, but such issues can be avoided altogether by having the prover provide Zero Knowledge Proofs of Knowledge for any ciphertext it would use as input. We discuss these risks in more detail in Sect. 3.1. We believe that, for many applications, the speed-ups we achieve through this paradigm shift far outweigh the one-bit privacy loss. This is reflected by the lack of implementations for works on the previous paradigm (verification over the ciphertext space), for which, except for very low-depth circuits, their lack of efficiency is a non-starter.

1.2 Comparison with Existing Works

In concurrent work [GGW23], Garg, Goel and Wang offer a framework to prove statements on values that are hidden from the prover. Their framework is based on making FRI work over such hidden values, and they show how to compile Polynomial IOPs into SNARKs given such a tool. Besides HE, their work considers more general ways to hide these values from the prover, such as homomorphic commitments and group exponentiation, grouped under the abstraction of Linearly-Homomorphic Encapsulations.

A formal issue in [GGW23] is that their notion of a *decryptable* (or that of *linearly-homomorphic w.r.t. randomness*) Linearly-Homomorphic Encapsulation is not sufficient when such an encapsulation is a building block of more complex components such as FRI or polynomial commitments. Namely, their notion only considers decryption of a freshly encrypted ciphertext on which no operations have been performed. This overlooks the fact that the evaluation correctness of HE schemes, which are based on (Ring) Learning with Errors, is function-specific and needs to support the operations computed within those components. Whereas FRI only requires to perform a series of linear combinations on the ciphertexts, it turns out that the size of the coefficients in the linear combination and the additive depth of FRI constitutes a significant obstacle for noise management in practice (see Sect. 6).

While [GGW23] is more theoretical and lacks implementation, our work focuses on concretely accelerating VC on encrypted data. In addition to various technical optimizations and trade-offs informed by our experimentation, we address two key aspects that [GGW23] does not: scenarios where the prover provides inputs in zero knowledge, and the direct use of FRI (e.g. by compiling δ -correlated IOPs into (zk)SNARKs [BGK+23]), rather than going through a polynomial commitment abstraction. The former greatly improves the parameters of the HE scheme in several applications, such as Privacy-Preserving Machine Learning (where the prover provides their model as a plaintext in zero-knowledge while the verifier’s input are HE ciphertexts), whereas the latter improves FRI’s parameters by allowing to use a proximity parameter up to the Johnson bound.

2 Preliminaries

We use $R[x]_{\leq d}$ to refer to polynomials with coefficients in a finite, commutative ring R and degree at most d . For an element $a \in R$, we write $[a]_q$ to denote the reduction of a modulo q (coefficient-wise), with the set of representatives of coefficients lying in $\{0, \dots, q-1\}$. This should not be confused with $[n]$, which we will sometimes use to denote the set of integers $\{1, \dots, n\}$.

We use bold notation (e.g. \mathbf{b}) to refer to vectors. We use $y \leftarrow C$ to denote that y is the output of a given computation C . We use $a \leftarrow A$ for sampling an element a from a distribution A . When A is a set rather than a distribution, we write $a \xleftarrow{\$} A$ for sampling a uniformly at random in A . We write $\llbracket f \rrbracket$ to denote an oracle to f , and M^f to denote that M has oracle access to f . We abbreviate Probabilistic Polynomial Time as PPT. We let λ denote a computational security parameter.

We denote computational indistinguishability by $\overset{c}{\approx}$ when no PPT algorithm can distinguish between two distributions except with negligible probability.

2.1 Basic Algebra and Galois Theory

Next, we present some Number and Galois Theory facts that were noted in the context of FHE in [SV14], but that are fairly standard. Let p be a prime, $F(x) \in \mathbb{F}_p[x]$ be a polynomial, $\deg(F) = N$, and assume that it factorises (mod p) into ℓ factors, all of degree D , for $D \cdot \ell = N$, i.e. $F(x) = \prod_{i=1}^{\ell} F_i(x) \pmod{p}$, where $\deg(F_i) = D, \forall i \in [\ell]$. Then we can define $R_p := \mathbb{F}_p[x]/(F)$, and $R_p \cong \mathbb{F}_p[x]/(F_1) \times \dots \times \mathbb{F}_p[x]/(F_{\ell}) \cong \mathbb{F}_{p^D} \otimes \dots \otimes \mathbb{F}_{p^D}$.

Let $F = \Phi_{2N}$ be the $2N^{\text{th}}$ cyclotomic polynomial, for N a power of two, and $\deg(\Phi_{2N}) = N$. We note in particular that the above implies that, if p is a prime such that \mathbb{F}_p contains a primitive $2\ell^{\text{th}}$ root of unity, we have that $\Phi_{2N}(x) = \prod_{i \in (\mathbb{Z}/2\ell\mathbb{Z})^\times} (x^D - \zeta^i) \pmod{p}$. In the *fully-splitting* case, we have that $D = 1$, and can therefore write R_p as the direct sum of N copies of \mathbb{F}_p . The cases where $D > 1$ is small with respect to N are *almost-fully splitting*.

2.2 Homomorphic Encryption

Definition 1 A public-key Homomorphic Encryption scheme HE over a set of admissible circuits $\widehat{\text{Circ}}$ consists of the following algorithms.

- $(\mathbf{pp}, \mathcal{C}) \leftarrow HE.Setup(1^\lambda, \mathcal{M}, \widehat{\text{Circ}})$: Given a message space \mathcal{M} , a set of admissible circuits $\widehat{\text{Circ}}$, output the public parameters \mathbf{pp} and the ciphertext space \mathcal{C} such that the scheme is semantically secure.
- $(\mathbf{sk}, \mathbf{pk}, \mathbf{evk}) \leftarrow HE.KeyGen(1^\lambda, \mathcal{C}, \mathbf{pp})$: Given the public parameters \mathbf{pp} and the ciphertext space \mathcal{C} , output the secret key \mathbf{sk} , the public key \mathbf{pk} and the evaluation key \mathbf{evk} .
- $\mathbf{ct} \leftarrow HE.Enc(\mathbf{pk}, m)$: Given a message $m \in \mathcal{M}$, output its encryption \mathbf{ct} .
- $m' \leftarrow HE.Dec(\mathbf{sk}, \mathbf{ct})$: Given a ciphertext \mathbf{ct} and its corresponding secret key \mathbf{sk} , output the decryption of \mathbf{ct} , m' .
- $\mathbf{ct}' \leftarrow HE.Eval(\mathbf{evk}, \mathbf{ct}, \hat{C})$: Given a circuit $\hat{C} \in \widehat{\text{Circ}}$, output the evaluation of \hat{C} on \mathbf{ct} .

When the context is clear, we will omit specifying the $\mathbf{sk}, \mathbf{pk}, \mathbf{pp}, \mathcal{C}$ parameters. The (standard) definition for semantic security of HE schemes is given in the full version [ACGS23].

Definition 2 Let HE be a homomorphic scheme as in Definition 1. We say that HE is correct if the equation $HE.Dec(\mathbf{sk}, HE.Eval(\mathbf{evk}, HE.Enc(\mathbf{pk}, m), \hat{C})) = C(m)$ holds with overwhelming probability for all admissible circuits $\hat{C} \in \widehat{\text{Circ}}$.

We note that the set $\widehat{\text{C}}$ refers to the set of circuits that the scheme can support – for example, some HE schemes support non-linear operations such as ReLu, and some do not. In particular, each circuit $\hat{C} \in \widehat{\text{Circ}}$ has a corresponding circuit C on the *plaintext space*. To give a concrete example, if we want to evaluate the homomorphic multiplication of two ciphertexts, \hat{C} could be a multiplication followed by a bootstrapping, whereas C would simply be a multiplication.

2.3 Reed Solomon Codes

Reed-Solomon (RS) codes are arguably the most common linear error correction codes. Let us recall their definition and fix some notation relative to them. In this work, we will parameterize them by a finite field \mathbb{F} , a multiplicative subgroup $L \subseteq \mathbb{F}^*$ and a degree bound d . Hence, $\text{RS}[\mathbb{F}, L, d]$ is defined as follows:

$$\text{RS}[\mathbb{F}, L, d] = \{(f(x))_{x \in L} \in \mathbb{F}^{|L|} : f \in \mathbb{F}[X]_{<d}\}.$$

The code rate of $\text{RS}[\mathbb{F}, L, d]$ is $\rho = d/|L|$. Unless we state it otherwise, in this work we will assume that $|L| = 2^k$, $\rho = 2^{-R}$ and $d = 2^{k-R}$.

For two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}^n$, we let $\Delta(u, v)$ denote the *relative Hamming distance* between \mathbf{u} and \mathbf{v} , defined as $\Delta(\mathbf{u}, \mathbf{v}) := |\{u_i \neq v_i \mid i \in \{1, \dots, n\}\}|/n$. For a set of vectors $S \subset \mathbb{F}^n$ and any vector $\mathbf{u} \in \mathbb{F}^n$, we define $\Delta(\mathbf{u}, S) = \Delta(S, \mathbf{u}) := \min_{\mathbf{v} \in S} \{\Delta(\mathbf{u}, \mathbf{v})\}$. For $\delta \in (0, 1)$, we say that \mathbf{u} is δ -far from S if $\Delta(\mathbf{u}, S) \geq \delta$. Otherwise, we say that \mathbf{u} is δ -close to S . Equivalently, \mathbf{u} is δ -far from S if $\Delta(\mathbf{u}, S) \geq \delta$ for all $\mathbf{v} \in S$, and \mathbf{u} is δ -close to S if there exists $\mathbf{v}^* \in S$ such that $\Delta(\mathbf{u}, \mathbf{v}^*) < \delta$. We refer to δ as the *proximity parameter*. When $\delta < (1 - \rho)/2$, we say that δ is within the *unique decoding radius*; and when $\delta < 1 - \sqrt{\rho}$, we say that δ is within the Johnson bound.

Definition 3 (Correlated agreement) *Let $\delta \in (0, 1)$. Let $V = \text{RS}[\mathbb{F}, L, d]$ and let $W = \{w_1, \dots, w_k\} \subseteq \mathbb{F}^{|L|}$. We say W has δ -correlated agreement with V on an agreement set $S \subseteq L$ if $|S|/|L| \geq 1 - \delta$ and there exist $v_1, \dots, v_k \in V$ such that, $\forall x \in S, w_i(x) = v_i(x)$.*

2.4 Interactive Oracle Proofs (of Proximity)

There are several variations of the IOP abstraction [BCS16]. Polynomial IOPs (PIOPs) ask for the IOP oracles to be polynomials evaluated over the entire field \mathbb{F} , whereas for the weaker notion of Reed Solomon-encoded IOPs (RS-IOPs) those are Reed-Solomon codewords (i.e. the evaluation of a polynomial over some specific domain $L \subset \mathbb{F}$). In this work, we focus on RS-encoded IOPs and on δ -correlated IOPs, which were introduced in [BGK+23]. Our results could nevertheless be easily adapted to other IOP flavors, e.g. to PIOPs as in [GGW23]. The main attractive of δ -correlated IOPs is that they allow for a better proximity parameter δ (up to the Johnson bound, rather than within the unique decoding radius) when they are compiled into SNARKs. When $\delta = 0$, δ -correlated IOPs can be seen as a subclass of RS-encoded IOPs [BCR+19, COS20].

Definition 4 *An indexed relation \mathcal{R} is a set of triples $(\mathbf{i}, \mathbf{x}; \mathbf{w}) \in \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^*$. The string \mathbf{x} is called *input, statement or instance*, the string \mathbf{w} is called the *witness* and the string \mathbf{i} is an *index*. The index can be thought as something that is fixed at setup time, and chooses among a universe of binary relations $\mathcal{R}_{\mathbf{i}} = \{(\mathbf{x}; \mathbf{w}) : (\mathbf{i}, \mathbf{x}; \mathbf{w}) \in \mathcal{R}\}$.*

In the setting of holographic proofs, a good example is an indexed relation for circuit satisfiability, where the index \mathbf{i} is a description of the circuit, the statement \mathbf{x} contains the “public” values on some of the circuit’s input wires and the witness \mathbf{w} consists in the values taken by the remaining “private” wires.

Definition 5 ([BGK+23]) *Let $H \subseteq \mathbb{F}$ and $d \geq 0$. An indexed (\mathbb{F}, H, d) -polynomial oracle relation \mathcal{R} is an indexed relation where for each $(\mathbf{i}, \mathbf{x}, \mathbf{w}) \in \mathcal{R}$, the index \mathbf{i} and input \mathbf{x} may contain oracles to codewords from $\text{RS}[\mathbb{F}, H, d]$ and the actual codewords corresponding to these oracles are contained in \mathbf{w} .*

Definition 6 *A μ -round holographic interactive oracle proof (hIOP) for an indexed relation \mathcal{R} is a tuple of PPT interactive algorithms $\Pi = (\mathcal{P}, \mathcal{V})$ and a deterministic polynomial-time algorithm Ind (the indexer), with two phases:*

- *In an offline phase, given an index \mathbf{i} , Ind computes an encoding of it, $\text{Ind}(\mathbf{i})$.*
- *In an online phase, $\mathcal{P}(\text{Ind}(\mathbf{i}), \mathbf{x}, \mathbf{w})$ and $\mathcal{V}^{\text{Ind}(\mathbf{i})}(\mathbf{x})$ exchange $2\mu + 1$ messages, where \mathcal{P} sends the first and last message. \mathcal{V} gets only oracle access to \mathcal{P} 's messages, and after \mathcal{P} 's final message, \mathcal{V} either accepts or rejects.*

Furthermore, an hIOP has to satisfy the two following properties:

Completeness: *For all $(\mathbf{x}, \mathbf{w}) \in \mathcal{R}$, we have that $\Pr[\langle \mathcal{P}(\text{Ind}(\mathbf{i}), \mathbf{w}), \mathcal{V}^{\text{Ind}(\mathbf{i})}(\mathbf{x}) \rangle = 1] \geq \gamma$, where the probability is taken over the random coins of \mathcal{V} . If, for all \mathbf{x} , $\gamma = 1$, then the hIOP has perfect completeness.*

Soundness: *For any $\mathbf{x} \notin \mathcal{L}_{\mathcal{R}}$ and any unbounded malicious \mathcal{P}^* , we have that $\Pr[\langle \mathcal{P}^*(\text{Ind}(\mathbf{i}), \mathbf{w}), \mathcal{V}^{\text{Ind}(\mathbf{i})}(\mathbf{x}) \rangle = 1] \leq \epsilon$, where the probability is taken over the random coins of \mathcal{V} .*

In δ -correlated hIOPs, the prover is supposed to send oracles to maps that agree with low degree polynomials on a fraction of $1 - \delta$ points (see Definition 3). On top of checking all the received oracles correspond indeed to δ -correlated maps (which we capture by the relation in Definition 7), it is necessary to verify some algebraic equalities involving some evaluations of those maps. These are made concrete in Definition 8.

Definition 7 ([BGK+23]) *Let $0 \leq \delta < 1$. The δ -correlated agreement relation for $\text{RS}[\mathbb{F}, L, d]$ is the following indexed (\mathbb{F}, L, d) -polynomial oracle relation:*

$$\text{CoAgg} = \left\{ \begin{array}{l} \left(\begin{array}{l} \mathbf{i} \\ \mathbf{x} \\ \mathbf{w} \end{array} \right) = \left(\begin{array}{l} (\mathbb{F}, L, d, \delta, r) \\ (\llbracket f_i \rrbracket)_{i \in [r]} \\ (f_i)_{i \in [r]} \end{array} \right) : \left. \begin{array}{l} r, \delta \geq 0, \rho = d/|L| \\ f_i \in \mathbb{F}^L \ \forall i \in [r] \\ (f_i)_{i \in [r]} \text{ has } \delta\text{-correlated agreement} \\ \text{with } \text{RS}[\mathbb{F}, L, d] \end{array} \right\}$$

Definition 8 (δ -correlated hIOP, [BGK+23]) *Let $L = \langle \omega \rangle$ be a smooth multiplicative subgroup of \mathbb{F}^* of order $d = 2^v / \rho$ for some $v \geq 1$ and rate $0 < \rho < 1$ and define the Reed-Solomon code $\text{RS}[\mathbb{F}, L, d]$. Let $0 \leq \delta < 1$ and let \mathcal{R} be an indexed (\mathbb{F}, L, d) -polynomial oracle relation. Let Π be a hIOP for \mathcal{R} . Given a (possibly partial) transcript (\mathbf{x}, τ) generated during Π , let $\text{Words}(\mathbf{x}, \tau)$ be the words from \mathbb{F}^L that fully describe the oracles appearing in (\mathbf{x}, τ) . We say that Π is δ -correlated if:*

- *The verifier \mathcal{V} has oracle access to the δ -correlated agreement relation $\text{CoAgg}(\delta)$.*

– For all $(\mathfrak{i}, \mathfrak{x}, \mathfrak{w}) \in \mathcal{R}$:

- In the last round of interaction between $\mathcal{P}(\text{Ind}(\mathfrak{i}), \mathfrak{x}, \mathfrak{w})$ and $\mathcal{V}^{\text{Ind}(\mathfrak{i}), \text{CoAgg}(\delta)}(\mathfrak{x})$, the verifier sends a field element z uniformly sampled from a subset of (a field extension of) \mathbb{F} and the honest prover replies with the values:

$$\text{Evals}(\mathfrak{x}, \tau, z) = (w(\omega^{k_w, 1} z), \dots, w(\omega^{k_w, n_w} z) : w \in \text{Words}(\mathfrak{x}, \tau))$$

where τ is the transcript so far and $\kappa = \{k_{w, i} : w \in \text{Words}(\mathfrak{x}, \tau), i \in [n_w]\}$ is a fixed set of integers which are output by Ind .

- To decide whether to accept or reject a proof, $\mathcal{V}^{\text{Ind}(\mathfrak{i}), \text{CoAgg}(\delta)}(\mathfrak{x})$ makes the two following checks:

Check 1 Assert whether the received values $\text{Evals}(\tau, z)$ are a root to some multivariate polynomial $F_{\mathfrak{i}, \mathfrak{x}, \tau}$ depending on \mathfrak{i} , \mathfrak{x} and τ .

Check 2 Assert whether the maps

$$\text{quotients}(\mathfrak{x}, \tau, z) = \left\{ \frac{w(\mathbf{X}) - w(\omega^{k_w, j} z)}{\mathbf{X} - \omega^{k_w, j} z} : w \in \text{Words}(\mathfrak{x}, \tau), j \in [n_w] \right\}$$

have δ -correlated agreement in $\text{RS}[\mathbb{F}, L, d - 1]$ by using the $\text{CoAgg}(\delta)$ oracle on the oracles to such maps.

Next, we define the notions of round-by-round (RBR) soundness and knowledge soundness [CCH+19] for holographic IOPs. Since those are a superset of δ -correlated hIOPs, the same definition applies to the latter.

Definition 9 A holographic IOP for an indexed relation \mathcal{R} has round-by-round (RBR) soundness with error ϵ if for every index \mathfrak{i} there exists a “doomed set” $\mathcal{D}(\mathfrak{i})$ of partial and complete transcripts such that:

1. If $\mathfrak{x} \notin \mathcal{L}_{\mathcal{R}, \mathfrak{i}}$, then $(\mathfrak{x}, \emptyset) \in \mathcal{D}(\mathfrak{i})$, where \emptyset denotes the empty transcript.
2. For every possible input \mathfrak{x} and complete transcript τ , if $(\mathfrak{x}, \tau) \in \mathcal{D}(\mathfrak{i})$, then $\mathcal{V}^{\text{Ind}(\mathfrak{i})}(\mathfrak{x}, \tau) = \text{reject}$.
3. If $i \in [\mu]$ and (\mathfrak{x}, τ) is a $(i-1)$ -round partial transcript such that $(\mathfrak{x}, \tau) \in \mathcal{D}(\mathfrak{i})$, then $\Pr_{c \leftarrow C_i} [(\mathfrak{x}, \tau, m, c) \notin \mathcal{D}(\mathfrak{i})] \leq \epsilon(\mathfrak{i})$ for every possible next prover message m .

Definition 10 A holographic IOP for an indexed relation \mathcal{R} has round-by-round (RBR) knowledge soundness with error ϵ_k if there exists a polynomial time extractor Ext and for every index \mathfrak{i} there exists a “doomed set” $\mathcal{D}(\mathfrak{i})$ of partial and complete transcripts such that:

1. For every possible input \mathfrak{x} (regardless of whether $\mathfrak{x} \notin \mathcal{L}_{\mathcal{R}, \mathfrak{i}}$ or not), $(\mathfrak{x}, \emptyset) \notin \mathcal{D}(\mathfrak{i})$.
2. For every possible input \mathfrak{x} and complete transcript τ , if $(\mathfrak{x}, \tau) \in \mathcal{D}(\mathfrak{i})$, then $\mathcal{V}^{\text{Ind}(\mathfrak{i})}(\mathfrak{x}, \tau) = \text{reject}$.
3. Let $i \in [\mu]$ and (\mathfrak{x}, τ) be a $(i-1)$ -round partial transcript such that $(\mathfrak{x}, \tau) \in \mathcal{D}(\mathfrak{i})$. If for every possible next prover message m it holds that $\Pr_{c \leftarrow C_i} [(\mathfrak{x}, \tau, m, c) \notin \mathcal{D}(\mathfrak{i})] > \epsilon_k(\mathfrak{i})$, then $\text{Ext}(\mathfrak{i}, \mathfrak{x}, \tau, m)$ outputs a valid witness for \mathfrak{x} .

Finally, let us also discuss zero knowledge, which will be particularly interesting in our work.

Definition 11 *An hIOP Π for an indexed relation \mathcal{R} has statistical zero knowledge with query bound b if there exists a PPT simulator \mathcal{S} such that for every $(\mathbf{i}, \mathbf{x}, \mathbf{w}) \in \mathcal{R}$ and any \mathcal{V}^* making less than b queries in total to its oracles, the random variables $\mathbf{View}(\mathcal{P}(\mathbf{i}, \mathbf{x}, \mathbf{w}), \mathcal{V}^*)$ and $\mathcal{S}^{\mathcal{V}^*}(\mathbf{i}, \mathbf{x})$, defined below, are statistically indistinguishable*

- $\mathbf{View}(\mathcal{P}(\mathbf{i}, \mathbf{x}, \mathbf{w}), \mathcal{V}^*)$ is the view of \mathcal{V}^* , i.e. the random variable (r, a_1, \dots, a_q) where r is \mathcal{V}^* randomness and a_1, \dots, a_q are the responses to \mathcal{V}^* 's queries determined by the oracles sent by \mathcal{P} .
- $\mathcal{S}^{\mathcal{V}^*}(\mathbf{i}, \mathbf{x})$ is the output of $\mathcal{S}(\mathbf{i}, \mathbf{x})$ when given straightline (i.e, without rewinding) access to \mathcal{V}^* , prepended with \mathcal{V}^* randomness r .

Π is honest-verifier zero knowledge if the above holds with $\mathcal{V}^* = \mathcal{V}^{\text{Ind}(\mathbf{i})}(\mathbf{x})$.

Some examples of δ -correlated hIOPs are Plonky2, RISC Zero, ethSTARK, Aurora and Fractal [COS20]. One particular advantage of hIOPs is how easy it is to compile them into SNARKs through the so-called BCS transformation [BCS16]. In a nutshell, this consists in replacing oracles sent by the prover with Merkle-tree-based commitments and then removing interaction with the verifier by applying the Fiat-Shamir transform. It has been proved that if an hIOP is round-by-round sound, applying the BCS transformation results in a SNARK that is adaptively knowledge sound versus both classic and quantum adversaries in the random oracle model [CMS19, COS20].

A similar concept to the above one is that of an IOP of Proximity (IOPP), which is an IOP to test proximity to a specific code. In this work, we restrict ourselves to IOPPs for Reed Solomon Codes.

Definition 12 *Let RS denote the family of Reed Solomon codes $\text{RS}[\mathbb{F}, L, d]$. A protocol between a pair of interactive machines $\langle \mathcal{P}, \mathcal{V} \rangle$ is an r -round interactive oracle proof of δ -proximity for RS is an IOP with the following modifications*

- **Input format:** The first message from \mathcal{P} is $f_0 : L \rightarrow \mathbb{F}$, allegedly a RS codeword.
- **Completeness:** $\Pr[\langle \mathcal{P}, \mathcal{V} \rangle = 1 : \Delta(f_0, \text{RS}) = 0] = 1 - \theta$ for a negligible θ . If $\theta = 0$ we refer to this as perfect completeness.
- **ϵ -soundness:** For any unbounded \mathcal{P}^* , $\Pr[\langle \mathcal{P}^*, \mathcal{V} \rangle = 1 : \Delta(f_0, \text{RS}) \geq \delta] \leq \epsilon$.

The next theorem summarizes the compilation results of [BGK+23]. Informally, given a δ -correlated hIOP, it suffices to analyse its RBR knowledge soundness when $\delta = 0$ and replace oracles with a δ -correlation check³ to produce an RBR knowledge sound hIOP as a result. This hIOP can then be turned into a SNARK through the usual BCS transformation [BCS16]. Interestingly, having $\delta \neq 0$ (and actually up to the Johnson bound!) does not affect knowledge soundness when following the [BGK+23] recipe, whereas previous compilers [CMS19, COS20] were restricted to the unique decoding regime, i.e. $\delta < (1 - \rho)/2$.

³ Such as batched FRI.

Theorem 1 ([BGK+23]). *Let $\Pi_\delta^\mathcal{O}$ be a δ -correlated hIOP, where \mathcal{O} is an oracle for δ -correlated agreement. Let $0 < \eta \leq 1$ and $\rho > 0$ be such that $\delta = 1 - \sqrt{\rho} - \eta$ is strictly positive. Assume $\Pi_0^\mathcal{O}$ has RBR knowledge soundness with error ϵ . Then, $\Pi_\delta^\mathcal{O}$ has RBR knowledge soundness with error $\epsilon/(2\eta\sqrt{\rho})$.*

Moreover if Π_{c_A} is an IOPP for δ -correlated agreement in $\mathbf{RS}[\mathbb{F}, L, d]$ with RBR soundness error ϵ_{c_A} , then the protocol $\Pi_\delta^{\Pi_{c_A}}$ obtained by replacing \mathcal{O} with Π_{c_A} in $\Pi_\delta^\mathcal{O}$ has RBR knowledge soundness error $\epsilon_1 = \max\{\epsilon/(2\eta\sqrt{\rho}), \epsilon_{c_A}\}$.

Furthermore, given a random oracle with λ -bit output and a query bound Q , compiling $\Pi_\delta^{\Pi_{c_A}}$ with the BCS transformation [BCS16] yields a SNARK with knowledge error $Q \cdot \max\{\epsilon/(2\eta\sqrt{\rho}), \epsilon_{c_A}\} + O(Q^2/2^\lambda)$.

3 Verifiable Computation over Encrypted Data

The notion of verifiable computation (VC) proposed by Gennaro et al. in [GGP10] tries to better capture the way in which proof and argument systems are used in practice. In Definition 1 we tweak their definition and syntax to fit our constructions. In particular, we allow for the verification algorithm to be interactive, since we will often discuss at the IOP rather than SNARK level of abstraction.

Furthermore, we would like to support circuits of the form $C(\mathbf{x}, \mathbf{w}_P)$ where (the homomorphic encryption of) the input \mathbf{x} is provided by the verifier, while the prover specifies \mathbf{w}_P , which includes plaintexts and/or ciphertexts revealed to \mathcal{V} during verification. Without threshold decryption to prevent the verifier from unauthorized decryptions, all \mathbf{w}_P values are exposed to the verifier. Thus, for single-client outsourcing, \mathbf{w}_P should consist only of plaintexts.

If \mathbf{w}_P must remain hidden, it can be treated as a private witness mixed with the encryption \mathbf{x} in zero-knowledge. For example, one could think about a private Machine-Learning-as-a-Service [AHH+24], where the client sends encrypted queries $\mathbf{HE}.\mathbf{Enc}(\mathbf{x})$ to the server, who applies their private model \mathbf{w}_P . This approach, also pursued in [BCFK21, GNS23], is advantageous because multiplying plaintexts (such as those in \mathbf{w}_P) with ciphertexts (the encryptions of \mathbf{x} and the outputs that result from operating on them) is much cheaper than multiplying ciphertexts.

Definition 1 (Verifiable Computation). *A verifiable computation scheme \mathbf{VC} is a tuple of polynomial time algorithms $(\mathbf{VC}.\mathbf{Setup}, \mathbf{VC}.\mathbf{ProbGen}, \mathbf{VC}.\mathbf{Compute}, \mathbf{VC}.\mathbf{Ver})$ defined as follows.*

- $(\mathbf{SK}, \mathbf{PK}) \leftarrow \mathbf{VC}.\mathbf{Setup}(1^\lambda, C)$: *A randomized key generation algorithm takes a circuit C as input and outputs a secret key \mathbf{SK} and a public key \mathbf{PK} .*
- $(\sigma_{\mathbf{x}}, \mathbf{VK}_{\mathbf{x}}) \leftarrow \mathbf{VC}.\mathbf{ProbGen}(\mathbf{PK}, \mathbf{x})$: *A randomized problem generation algorithm (to be run by \mathcal{V}) takes the public key \mathbf{PK} , an input \mathbf{x} , and outputs a public encoding $\sigma_{\mathbf{x}}$ of \mathbf{x} , together with a private verification key $\mathbf{VK}_{\mathbf{x}}$.*
- $\eta_{\mathbf{y}} \leftarrow \mathbf{VC}.\mathbf{Compute}(\mathbf{PK}, \sigma_{\mathbf{x}}, \mathbf{w}_P, C)$: *Given a public key \mathbf{PK} for a circuit C , the encoded input $\sigma_{\mathbf{x}}$ and input \mathbf{w}_P , \mathcal{P} computes $\eta_{\mathbf{y}}$, which consists of an encoded version $\sigma_{\mathbf{y}}$ of the circuit's output $\mathbf{y} = C(\mathbf{x}, \mathbf{w}_P)$ and data to answer challenges about that statement.*

- $\text{acc} \leftarrow \text{VC.Ver}(\mathcal{P}(\text{PK}, \eta_y), \mathcal{V}(\text{SK}, \text{VK}_x, \sigma_y))(C)$: The interactive verification algorithm uses the input-specific verification key VK_x , the setup secret key SK and a proof η_y to return σ_y together with a bit $\text{acc} \in \{0, 1\}$ such that $\text{acc} = 1$ if $\text{VC.Decode}(\sigma_y, \text{SK}) = C(x, w_P)$ or $\text{acc} = 0$ otherwise.
- $y \leftarrow \text{VC.Decode}(\sigma_y, \text{SK})$: Using the secret key, the decoding algorithm outputs the value y behind the public encoding σ_y .

A verifiable computation scheme can satisfy a range of properties which we next define. We omit the VC. prefix in the different algorithms for ease of readability. In our work, we will always be interested in all of the following ones whenever w_P does not need to be kept private.

- *Correctness.* Correctness guarantees that if \mathcal{P} is honest, the verification test will pass. That is, for all C , and for all valid inputs x, w_P of C the following probability equals $1 - \text{negl}(\lambda)$.

$$\Pr \left(\begin{array}{l} \text{acc} = 1 \\ \text{Decode}(\sigma_y, \text{SK}) = C(x, w_P) \end{array} : \begin{array}{l} (\text{SK}, \text{PK}) \leftarrow \text{Setup}(1^\lambda, C) \\ (\sigma_x, \text{VK}_x) \leftarrow \text{ProbGen}(\text{PK}, x) \\ \eta_y \leftarrow \text{Compute}(\text{PK}, \sigma_x, w_P, C) \\ \text{acc} \leftarrow \text{Ver}(\mathcal{P}(\text{PK}, \eta_y), \mathcal{V}(\text{SK}, \text{VK}_x, \sigma_y))(C) \end{array} \right)$$

- *Outsourceability.* A VC scheme is outsourceable if for any x and any η_y , the time required by \mathcal{V} to run $\text{ProbGen}(x)$, $\text{Ver}(\mathcal{P}(\text{PK}, \eta_y), \mathcal{V}(\text{SK}, \text{VK}_x))(C)$ and $\text{Decode}(\sigma_y, \text{SK})$ is $o(T)$, where T is the time required to compute $C(x, w_P)$.
- ϵ -*Soundness.* A VC scheme is ϵ -sound if the advantage of any PPT adversary \mathcal{A} in the game $\text{Exp}_A^{\text{Ver}}$ defined as $\Pr [\text{Exp}_A^{\text{Ver}}[VC, C, \lambda] = 1]$ is ϵ .
- *Verifier-Privacy.* For a valid C , $\Pr [\text{Exp}_A^{\text{V.Priv}}[VC, C, \lambda] = 1] \leq 1/2 + \text{negl}(\lambda)$.

<pre> 1 Game $\text{Exp}_A^{\text{Ver}}(VC, C, \lambda)$ 2 $(\text{SK}, \text{PK}) \leftarrow \text{Setup}(1^\lambda, C)$ 3 $x \leftarrow \mathcal{A}(\text{PK}, C)$ 4 $(\sigma_x, \text{VK}_x) \leftarrow \text{ProbGen}(\text{PK}, x)$ 5 $\eta_y \leftarrow \mathcal{A}(\text{PK}, \sigma_x, C)$ 6 $\text{acc} \leftarrow$ 7 $\text{Ver}(\mathcal{P}(\text{PK}, \eta_y), \mathcal{V}(\text{SK}, \text{VK}_x, \sigma_y))(C)$ 8 $y \leftarrow \text{Decode}(\sigma_y, \text{SK})$ 9 if $\text{acc} = 1 \wedge \exists w_P : C(x, w_P) = y$ 10 return 1 </pre>	<pre> 1 Game $\text{Exp}_A^{\text{V.Priv}}(VC, C, \lambda)$ 2 $b \stackrel{\\$}{\leftarrow} \{0, 1\}$ 3 $(\text{SK}, \text{PK}) \leftarrow \text{Setup}(1^\lambda, C)$ 4 $(x_0, x_1, \text{state}) \leftarrow \mathcal{A}(\text{PK}, C)$ 5 $(\sigma_{x_b}, \text{VK}_{x_b}) \leftarrow \text{ProbGen}(\text{PK}, x_b)$ 6 $\hat{b} \leftarrow \mathcal{A}(\text{state}, \sigma_{x_b})$ 7 return $b \stackrel{?}{=} \hat{b}$ </pre>
--	--

3.1 On Verifier Privacy and Oracle Attacks

Our definition of verifier-privacy assumes that the verifier does not signal to the prover whether verification passes or not (i.e. there are no verification oracles) and that the value y it obtains after running the Decode algorithm is not used

as input to future algorithms (which we refer to as a decryption oracle). We briefly discuss the privacy impact of such oracles, which is inherent to moving verification from the ciphertext to the plaintext layer.

Verification oracles: When a malicious prover provides tampered ciphertexts (e.g. by manipulating their noise), the result of decryption is a function of such changes. The results of decryption also depend on the secret key and underlying message. Since the actions of the verifier (such as rejecting a proof or not, i.e. the value $\text{acc} \in \{0, 1\}$) depend on the decrypted ciphertexts, this is a source of leakage. All protocols we present are non-interactive, implying that there is at most one such verification oracle per proof sent. Hence, a malicious prover can learn at most a one-bit leakage predicate (which evaluates to acc) about the secret key/message. Regarding leakage about the key, the concrete security loss this incurs can be measured with tools such as the Leaky Estimator [DDGR20]. Additionally, stopping any further executions with a prover for which $\text{acc} = 0$, refreshing the keys and/or increasing their length are easy solutions.

Decryption Oracles: First, note that the output of the decryption of the prover's answers to verifier challenges has no use beyond verification. Therefore, the only value we need to be concerned about in a decryption oracle is the output of the computation itself. Consequently, decryption oracles may only occur in the composability case – i.e., when $\text{VC.Decode}(\sigma_y, \text{SK})$ is used as input in another protocol. Further, these decryption oracles arise only in the case where verification passes, as the Verifier would otherwise abort (and then, at worst, we would be in the presence of a verification oracle, as explained above). Finally, we note that the output of the computation ($C(\mathbf{x}, w_{\mathcal{P}})$) leaks nothing unexpected, as long as $w_{\mathcal{P}}$ does not contain ciphertexts for which \mathcal{P} did not provide a zero knowledge proof of knowledge (ZKPoK) for the underlying plaintext. In summary, the only scenario in which a decryption oracle may pose a threat is: the prover provides as input ciphertext without an accompanying ZKPoK, verification passes and the HELIOPOLIS output is passed along to another protocol for further processing.

3.2 Prover Privacy

In previous works on verifiable computation over encrypted data, the goal of keeping $w_{\mathcal{P}}$ private was modeled as a *context-hiding* property of the VC scheme [BCFK21, GNS23]. This is a reminiscence of a similar notion in the setting where $w_{\mathcal{P}}$ did not exist, but the parties running VC.ProbGen and VC.Ver were different [FNP20]. In our following sections we will deviate from that modeling and hence refrain from the context-hiding property. We do this not only because we focus on the more common scenario where the verifier is running both VC.ProbGen and VC.Ver , but also because context-hiding would not be able to model e.g. the interactivity of VC.Ver . We believe that our new security modeling will be useful for future work in this area.

We formalize the notion of *honest-verifier prover privacy* (HVPP) by showing that whatever a semi-honest \mathcal{V} can compute by participating in the protocol, \mathcal{V} could compute merely from its input and prescribed output. Our definition is in

the simulation paradigm and thus we have a stateful simulator \mathcal{S} that generates \mathcal{V} 's view given its input and output. We remark that, since \mathcal{V} is semi-honest, it is guaranteed that it uses its actual input and random tapes. In particular, \mathcal{S} can furthermore generate \mathcal{V} 's random tape and, at that point, generate the whole protocol transcript on its own without ever needing to interact with \mathcal{V} .

Definition 2. *We say that a VC protocol is honest-verifier prover-private (HVPP) if there exists a PPT simulator \mathcal{S} such that for every circuit C :*

$$\{\mathcal{S}(1^\lambda, \text{PK}, \mathbf{x}, C(\mathbf{x}, \mathbf{w}_{\mathcal{P}}))\}_{\mathbf{x}, \mathbf{w}_{\mathcal{P}}, \lambda, \text{PK}} \stackrel{c}{\approx} \{\text{View}_{\mathcal{V}}(\text{SK}, \text{PK}, \mathbf{x}, \mathbf{w}_{\mathcal{P}}, \lambda)\}_{\mathbf{x}, \mathbf{y}, \lambda, \text{SK}, \text{PK}}$$

where $(\text{SK}, \text{PK}) \leftarrow \text{VC.Setup}(1^\lambda, C)$ and $\text{View}_{\mathcal{V}}(\text{SK}, \text{PK}, \mathbf{x}, \mathbf{w}_{\mathcal{P}}, \lambda)$ denotes the view of \mathcal{V} during an execution of the protocol on inputs $(\mathbf{x}, \mathbf{w}_{\mathcal{P}})$ and security parameter λ , that is $(\text{SK}, \text{PK}, \mathbf{x}, \mathbf{r}; m_1, \dots, m_e, \text{out})$ where \mathbf{r} is \mathcal{V} 's random tape, each m_i value is the i -th message \mathcal{V} receives and out denotes \mathcal{V} 's output, which is computed from all other values in its own view of the execution.

In Fig. 1, we provide our general recipe for a correct, sound and verifier-private Verifiable Computation scheme. Verifier-privacy follows from the use of encryption within the ProbGen step, and correctness from the fact that such encryption is homomorphic. Soundness is less immediate, since it requires to have an HE-IOP at hand, which is an object we define and construct in Sect. 4. Notice that, since we specialize the construction to use *holographic* IOPs (as a means to achieve outsourceability), the syntax of the verification is slightly modified, replacing the circuit C with the corresponding indexer algorithm.

4 Compiling Interactive Oracle Proofs to Work over HE

Given an IOP which was not conceived to work over encrypted data, we show how to adapt it to work with HE in Definition 13.

Definition 13 (HE-transformation) *Let $\langle \mathcal{P}(\mathbf{x}, \mathbf{w}), \mathcal{V}(\mathbf{x}) \rangle$ be an IOP, where the elements of \mathbf{x} and \mathbf{w} belong to a finite field \mathbb{F} . We define its encrypted version HE-IOP, for some HE scheme as follows:*

- There is a trusted setup $(\mathbf{pp}, \mathcal{C}) \leftarrow \text{HE.Setup}(1^\lambda, R_p, \widehat{\text{Circ}})$, where R_p splits into copies of \mathbb{F} and $\widehat{\text{Circ}}$ is a family of admissible circuits that captures all necessary computation within the IOP as well as any preceding/posterior one (such as coming up with parts of the witness, or using an IOPP and the BCS transformation to compile into a SNARK).
- There is also a trusted key generation step $(\mathbf{sk}, \mathbf{pk}, \mathbf{evk}) \leftarrow \text{HE.KeyGen}(1^\lambda, \mathcal{C}, \mathbf{pp})$. \mathcal{P} has as an additional input \mathbf{evk} and \mathcal{V} has as an additional input \mathbf{sk} .
- \mathcal{P} 's input \mathbf{x} is replaced by its encryption $\text{HE.Enc}(\mathbf{x})$. Parts of \mathbf{w} could be also replaced by their homomorphic encryption.
- As a result, some oracles in the HE-IOP might now contain ciphertexts. We refer to them as HE-oracles or encrypted oracles. \mathcal{V} has to decrypt the ciphertexts obtained from HE-oracles and perform the same checks as in \mathbb{F} .

Verifiable Computation over encrypted data

Let IOP be an holographic Interactive Oracle Proof. Let HE be an exact homomorphic encryption scheme with plaintext space R_p . Let $\phi : R_p \rightarrow \bigotimes_{i=1}^{\ell} \mathbb{F}_{p^D}$ be the CRT isomorphism. Let $C : \mathbb{F}_p^{inp} \times \mathbb{F}_p^{wit} \rightarrow \mathbb{F}_p^{out}$ be an arithmetic circuit that we want to verify. Let $L_x = \lceil inp/\ell \rceil$, $L_w = \lceil wit/\ell \rceil$.

$(\mathbf{sk}, (\mathbf{pk}, \mathbf{evk})) \leftarrow \text{Setup}(1^\lambda, C)$: Run HE setup:

1. \mathcal{V} determines a set of admissible circuits \widehat{Circ} which contains a circuit \widehat{C} that homomorphically computes C .
2. \mathcal{V} runs $(\mathbf{pp}, R_q^2) \leftarrow \text{HE.Setup}(1^\lambda, R_p, \widehat{Circ})$.
3. \mathcal{V} runs $(\mathbf{sk}, \mathbf{pk}, \mathbf{evk}) \leftarrow \text{HE.KeyGen}(1^\lambda, C, \mathbf{pp})$.
4. Given an index i for the circuit C , the indexer Ind computes an encoding of it, $\text{Ind}(i)$.

$(\sigma_x, \mathbf{vk}_x) \leftarrow \text{ProbGen}(\mathbf{x}, \mathbf{pk})$: \mathcal{V} parses $\mathbf{x} = (x_0, \dots, x_{inp-1}) \in \mathbb{F}_p^{inp}$. For $i = 0, \dots, L_x - 1$, let $m_{x,i} = \phi^{-1}(x_{i\ell}, \dots, x_{(i+1)\ell-1})$. Encrypt these inputs as $\sigma_x = \{\text{HE.Enc}(\mathbf{pk}, m_{x,i})\}_{i=0}^{L_x-1}$. Set $\mathbf{vk}_x = (\mathbf{sk}, \mathbf{x})$.

$\mathbf{w} \leftarrow \text{Compute}(\mathbf{evk}, \sigma_x, \mathbf{w}_P, \widehat{C})$: \mathcal{P} parses σ_x and evaluates $\widehat{C}(\sigma_x, \mathbf{w}_P)$, by which it obtains the rest of the witness: the values on intermediate wires \mathbf{w}_C and the circuit output σ_y . Notice that the whole witness becomes $\mathbf{w} = (\mathbf{w}_P, \mathbf{w}_C, \sigma_y)$, which is a mix of plaintext values (such as \mathbf{w}_P) and ciphertext values (those depending on any input σ_x).

$\text{acc} \leftarrow \text{Ver}(\mathcal{P}(\text{Ind}(i), \mathbf{evk}, (\sigma_x, \mathbf{w})), \mathcal{V}^{\text{Ind}(i)}(\mathbf{sk}, (\mathbf{vk}_x, \sigma_y)))$: \mathcal{P} and \mathcal{V} run the HE-IOP corresponding to the IOP for the plaintext circuit C , i.e. $\langle \mathcal{P}(\text{Ind}(i), \mathbf{evk}, \sigma_x, \mathbf{w}), \mathcal{V}^{\text{Ind}(i)}(\mathbf{sk}, (\mathbf{x}, \text{HE.Dec}(\mathbf{sk}, \sigma_y))) \rangle$.

$\mathbf{y} \leftarrow \text{Decode}(\sigma_y, \mathbf{sk})$: \mathcal{V} outputs $\mathbf{y} = \text{HE.Dec}(\mathbf{sk}, \sigma_y)$.

Fig. 1. Verifiable Computation over Encrypted Data through HE-IOPs.

If we need to refer explicitly to the HE-IOP, we denote it as $\langle \mathcal{P}(\mathbf{evk}, \text{HE.Enc}(\mathbf{x}), \mathbf{w}), \mathcal{V}(\mathbf{sk}, \mathbf{x}) \rangle$, as a slight abuse of notation of the original \mathcal{P} and \mathcal{V} . The different properties of IOPs (completeness, soundness, round-by-round soundness, round-by-round knowledge soundness) can be trivially redefined for HE-IOPs.

One of the main interests of our HE-transformation, besides its simplicity, is that it preserves most parameters of the original IOP, with only some negligible degradation due to the use of homomorphic encryption.

Theorem 2. *Let IOP be an ϵ_k RBR knowledge sound, ϵ_{rbr} RBR sound, ϵ -sound, complete IOP. It's encrypted version HE-IOP is $\epsilon_k + \text{negl}(\lambda)$ RBR knowledge sound, $\epsilon_{rbr} + \text{negl}(\lambda)$ RBR sound, $\epsilon + \text{negl}(\lambda)$ -sound and complete.*

Proof. Completeness follows from the evaluation correctness of the HE scheme and the way the circuit family \widehat{Circ} was chosen. There is only a negligible loss in γ due to the way evaluation correctness is defined (Definition 2).

The soundness, RBR soundness and RBR (knowledge) soundness of an HE-IOP can be reduced to that of IOP as follows. Let \mathcal{A} be an adversary against HE-IOP with an advantage bigger than adding a factor $\text{negl}(\lambda)$ to the one for

the corresponding notion of the IOP ($\epsilon, \epsilon_{\text{rbr}}, \epsilon_k$ respectively). We will build an adversary \mathcal{A}' against IOP with the same such greater advantage and hence reach a contradiction. \mathcal{A}' runs $(\text{sk}, \text{pk}, \text{evk}) \leftarrow \text{HE.KeyGen}(1^\lambda, \mathcal{C}, \text{pp})$ for the relevant HE scheme, obtaining in particular sk . Given any input, \mathcal{A}' encrypts it under pk and forwards it to \mathcal{A} . For every message received from \mathcal{V} , \mathcal{A}' directly forwards it to \mathcal{A} . In order to reply to those, \mathcal{A}' queries the encrypted oracles $\llbracket f \rrbracket_{\text{HE}}$ received from \mathcal{A} at every point, decrypts the answers using sk to recover f , and then sends the oracle $\llbracket f \rrbracket$ to \mathcal{V} . Clearly, if \mathcal{A} succeeds, so does \mathcal{A}' . ■

The HE-transformation applies to all variants of IOPs presented in this paper, such as holographic IOPs, RS-encoded hIOPs, δ -correlated hIOPs and IOPs of proximity. In order to denote this transformation, we will also add the HE prefix to those (HE-hIOPs, δ -correlated HE-hIOPs, HE-IOPP, etc.).

The upcoming subsections are organized as follows. In Sect. 4.1, we discuss how to keep w hidden from the verifier through zero knowledge. Sections 4.2 and 4.3 show how to compile these HE-IOPs into HE-friendly SNARKs using an HE-friendly low degree test (such as the HE transformation of the Batched FRI protocol, which we will show in Sect. 5.1).

4.1 Achieving Prover-Privacy from ZK-IOPs

We first consider the case of honest-verifier prover-privacy (HVPP, see Definition 2), since it allows for a more practical construction and it also acts as a stepping stone towards understanding the malicious case. There are three main aspects to consider when compiling using IOPs for a prover-private version of Fig. 1, which we describe next. Two of them (Consideration #1 and #3) are specific to the use of homomorphic encryption.

Consideration #1: Circuit Privacy. A requirement for prover-private constructions is the fact that the HE scheme needs to support *circuit-privacy*. Namely, all the ciphertexts of the HE-IOP that are exposed to the verifier need to be re-randomized, since their noise carries information about the circuit that was computed on them and hence⁴ about $w_{\mathcal{P}}$. We provide our own definition of the circuit-privacy notion that is best aligned with our HVPP goal.

Definition 14 *A homomorphic encryption scheme HE (see Definition 1) is circuit-private if there exists a rerandomization algorithm $\text{HE.Rerand}(\text{evk}, \text{pk}, \hat{C}, \text{ct})$ and a simulator \mathcal{S}_{HE} such that, for any admissible circuit \hat{C} with inputs $\text{HE.Enc}(x_1), \dots, \text{HE.Enc}(x_n)$ and outputs $\text{ct}_{y_1}, \dots, \text{ct}_{y_m}$, it holds that (some inputs omitted for simplicity): $(\text{sk}, \text{Rerand}(\text{ct}_{y_1}), \dots, \text{Rerand}(\text{ct}_{y_m})) \stackrel{c}{\approx} (\text{sk}, \mathcal{S}_{\text{HE}}(\text{pk}, C(x_1, \dots, x_n)))$.*

One of the standard ways that the above definition can be achieved is by employing noise flooding to instantiate HE.Rerand , as done in [Gen09a]. In more

⁴ Notice that one can think about the circuit evaluation $C(x, w_{\mathcal{P}})$ with a private $w_{\mathcal{P}}$ as providing the evaluation of some unspecified circuit from the family $\{C_{w_{\mathcal{P}}}(x)\}_{w_{\mathcal{P}}}$.

detail, we add to the verifier-exposed ciphertexts an encryption of 0 with large enough noise to statistically hide the noise of the circuit that led to the production of that specific ciphertext. We will denote by $\Omega_{0,C}$ the set of such encryptions of zero. Notice that since all messages within an encrypted oracle are susceptible of being queried, we need to add such an encryption of zero to each of them *before* putting them within the oracle.

Consideration #2: Combining zkIOPs with LDTs. Assume either a zero-knowledge RS-hIOP or a δ -correlated hIOP is given. To compile it into a zk-IOP while making black-box use of a pre-existing Low Degree Test (in the form of an IOPP such as FRI), it is necessary for the prover to additionally send a random codeword r ahead of time, which is added to the linear combination of functions that are being tested for low-degreeness. Adding such an r does not affect soundness. However, since input to the LDT is now a *random* codeword, there is no need to worry about its inner workings beyond knowing what is the amount of queries made to the random codeword (which links with Consideration #3). For a more detailed leakage analysis when using FRI, see [Hab22].

Consideration #3: Combining zkIOPs with LDTs – Query Blow-Up from Packing. Compilers, such as the ones discussed in Consideration #2 and the one presented in Sect. 4.2, incur losses in several parameters of the resulting output IOP according to the number of queries to the LDT. This includes soundness, which in turn also loops into increasing the size of the underlying field in order to compensate. But, most importantly, the increase in the number of queries through the introduction of the LDT also degrades the query bound for zero knowledge (see Definition 11). As an example, see [COS20, Theorem 8.1.]

To make things worse, the HE-transformation of these protocols replaces oracles with *encrypted oracles*, where ciphertexts (rather than plaintexts) are placed within them. This means that, if the chosen HE scheme supports plaintext packing and we are exploiting this property, whenever the verifier \mathcal{V} would need to query only one of the plaintexts m_i on the ciphertext $\text{ct} = \text{HE.Enc}(m_1, \dots, m_l)$ behind the oracle, \mathcal{V} learns every other plaintext $m_j, j \neq i$ within it. Effectively, this blows-up the query loss for zero knowledge by a factor of up to l .⁵ Packing becomes then as devastating (or even more) for zero knowledge as it is an improvement for computational efficiency, which is a very problematic tension in practice. Hence, it is paramount to reduce the packing-induced multiplicative loss while maintaining efficiency. We provide a solution for this in Sect. 6.4.

Malicious Verifier We will only briefly address handling a malicious verifier. Instead of creating an ad-hoc “malicious-verifier prover-private” notion, it is best to model security as a maliciously secure 2-party computation protocol (see [CCL15]). Beyond the precautions for an honest verifier, we must ensure honest behavior in the Setup and ProbGen steps. This can be achieved through a trusted setup and enforced via zero-knowledge proofs, similar to a GMW-style compiler from passive to active security [GMW87].

⁵ It could be that \mathcal{V} sometimes happens to query values that happen to be packed within the same ciphertext, slightly reducing the blow-up in this case.

For the ProbGen step, it is crucial to ensure the verifier provides valid ciphertexts, meaning the noise must be within bounds in lattice-based HE. Using a zero-knowledge proof of knowledge (ZKPoK) ensures the right bounds for the cleartext and encryption randomness (see e.g. [DPSZ12, Figure 9]).

4.2 A Compiler for RS-Encoded IOPs

Our first compiler is for Reed-Solomon encoded IOPs, and is a result of adapting the works of Aurora [BCR+19] and Fractal [COS20].

Protocol 1 (Aurora/Fractal) Let $(\mathcal{P}_{\mathcal{R}}(\mathbb{x}, \mathbb{w}), \mathcal{V}_{\mathcal{R}}(\mathbb{x}))$ be an RS-encoded hIOP over $L \subseteq \mathbb{F}$, with maximum degree (d_c, d_e) for an indexed relation \mathcal{R} . Let HE-IOP be its HE-transformation. Let $(\mathcal{P}_{LDT}, \mathcal{V}_{LDT})$ be an IOPP for the RS code $\text{RS}[\mathbb{F}, L, d_c]$ with proximity parameter $\delta < \min(\frac{1-2\rho_c}{2}, \frac{1-\rho_c}{3}, 1-\rho_e)$ where $\rho_c = (d_c + 1)/|L|$ and $\rho_e = (d_e + 1)/|L|$. Let HE-IOPP be its HE-transformation. Proceed as follows:

1. **Masking codeword for low-degree test:** \mathcal{P} sends \mathcal{V} an oracle to a random $r \in \text{RS}[\mathbb{F}, L, d_c]$. This step can be skipped when not interested in obtaining a zk -HE-hIOP.
2. **RS-encoded HE-IOP for \mathcal{R} :** In parallel to the above, \mathcal{P} and \mathcal{V} simulate $(\mathcal{P}_{\mathcal{R}}(\text{HE.Enc}(\mathbb{x}), \mathbb{w}), \mathcal{V}_{\mathcal{R}}(\mathbb{x}))$. Over the course of this protocol, the prover sends encrypted oracles containing codewords $\pi_1 \in \text{RS}[\mathbb{F}, L, \mathbf{d}_1], \dots, \pi_{k^{\mathcal{R}}} \in \text{RS}[\mathbb{F}, L, \mathbf{d}_{k^{\mathcal{R}}}]$, and the verifier specifies a set of rational constraints \mathfrak{C} [COS20, Definition 4.1]. Let $l := \sum_{i=1}^{k^{\mathcal{R}}} l_i + |\mathfrak{C}|$.
3. **Random linear combination:** \mathcal{V} samples $\mathbf{v} \in \mathbb{F}^{2l}$ uniformly at random and sends it to \mathcal{P} .
4. **Low-degree test through HE-IOPP:** \mathcal{P} and \mathcal{V} simulate $(\mathcal{P}_{LDT}(\mathbf{v}^{\top} \Pi + r), \mathcal{V}_{LDT}^{\mathbf{v}^{\top} \Pi + r})$, where $\Pi := \begin{pmatrix} \Pi_0 \\ \Pi_1 \end{pmatrix} \in \mathbb{F}^{2l \times L}$ is defined as in [BCR+19, Protocol 8.2].
5. \mathcal{V} accepts if and only if \mathcal{V}_{LDT} accepts

Theorem 3. Protocol 1 is an HE-hIOP for \mathcal{R} with the following parameters, where the \mathcal{R} (resp. LDT) superscript denotes the parameters of the RS-encoded IOPP (resp. IOPP). Round complexity: $k^{\mathcal{R}} + k^{LDT}$. Query complexity: $q_{\pi}^{LDT} + q_{\mathbb{w}}^{LDT}(k^{\mathcal{R}} + 1)$. Proof length $\text{HE.Expand}(L^{\mathcal{R}} + L^{LDT})$, where HE.Expand is a ciphertext expansion function that depends on the specific HE scheme and how the different intermediate values are computed. Round-by-round soundness error: $\epsilon_1 = \max(\epsilon_{\text{rbr}}^{\mathcal{R}}, \epsilon_{\text{rbr}}^{LDT}, |L|/|\mathbb{F}|)$. Round-by-round knowledge error: $\epsilon_2 = \max(\epsilon_{\text{knw}}^{\mathcal{R}}, \epsilon_{\text{rbr}}^{LDT}, |L|/|\mathbb{F}|)$. Furthermore, if the RS-encoded IOP is zero-knowledge, then so is Protocol 1, with the same query bound.

Proof. All the claimed parameters can be reduced to the ones claimed in [COS20, Theorem 8.2]. The round and query complexity clearly remain the same as in there, and the proof length is only affected by the ciphertext expansion of the HE scheme. Completeness and RBR (knowledge) soundness follow from Theorem 2 and [COS20, Theorem 8.2]. \blacksquare

4.3 A Correlated-Agreement-Based Compiler

Our second compiler allows to set the proximity parameter up to the Johnson bound, which improves efficiency. It is the result of adapting one of the central theorems in [BGK+23] through the application of the HE transformation (Definition 13). The overall compiler appears in Fig. 2.

Theorem 4. *Let $\Pi_\delta^\mathcal{O}$ be a δ -correlated HE-hIOP, where \mathcal{O} is an HE-oracle for δ -correlated agreement in $\text{RS}[\mathbb{F}, L, d]$. Let $0 < \eta \leq 1$ and $\rho > 0$ be such that $\delta = 1 - \sqrt{\rho} - \eta$ is strictly positive. Assume $\Pi_0^\mathcal{O}$ has RBR knowledge soundness with error ϵ . Then, $\Pi_\delta^\mathcal{O}$ has RBR knowledge soundness with error $\epsilon/(2\eta\sqrt{\rho})$.*

Let HE be an homomorphic encryption scheme whose plaintext space R_p splits into copies of \mathbb{F} . If $\Pi_{\text{HE-CA}}$ is an HE-IOPP for δ -correlated agreement in $\text{RS}[\mathbb{F}, L, d]$ with RBR soundness error ϵ_{CA} , then the protocol $\Pi_\delta^{\Pi_{\text{HE-CA}}}$ obtained by replacing \mathcal{O} with $\Pi_{\text{HE-CA}}$ in $\Pi_\delta^\mathcal{O}$ has RBR knowledge soundness error $\epsilon_1 = \max\{\epsilon/(2\eta\sqrt{\rho}), \epsilon_{\text{CA}}\}$.

Furthermore, given a random oracle with λ -bit output and a query bound Q , compiling $\Pi_\delta^{\Pi_{\text{HE-CA}}}$ with the BCS transformation [BCS16] yields a SNARK (over encrypted data) with knowledge error $Q \cdot \max\{\epsilon/(2\eta\sqrt{\rho}), \epsilon_{\text{CA}}\} + O(Q^2/2^\lambda)$.

Proof. Consequence of combining Theorems 2 and 1. ■

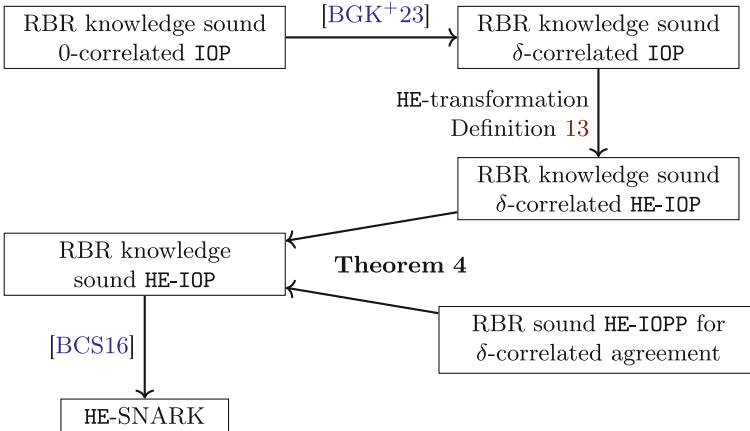


Fig. 2. Summary of compilation flow for δ -correlated IOPs.

5 Low Degree Tests for Encrypted Polynomials

The compilers from Sect. 4 need to eventually test whether the oracles sent by the IOP prover correspond to low-degree polynomials or not, with different variations

of what such a test should exactly verify (δ -correlated agreement or merely closeness to an RS code). First of all, we need to think about how polynomials mix with HE. For example, the following map

$$f : R_p \rightarrow R_p$$

$$a \mapsto \text{HE.Dec}\left(\sum_{i=0}^d \text{ct}_i \cdot a^i\right), \quad \text{ct}_i = \text{HE.Enc}(f_i)$$

only corresponds to a degree- d polynomial $f \in R_p[\mathbf{X}]$ as long as $f(a) = \sum_{i=0}^d f_i a^i \forall a \in R_p$, i.e. as long as it preserves evaluation correctness⁶.

In this work, as it is common in the IOP literature, polynomials are given in a point-value representation, which matches the definition of a Reed Solomon codeword. There are a series of operations that the prover (and maybe the verifier) will have to perform on the ciphertexts within those oracles, so we also need to make sure to preserve evaluation correctness when presented with such a representation. To achieve this, we introduce the notion of *encrypted polynomials*.

Definition 15 *Let HE be a homomorphic encryption scheme with plaintext space $R_p \cong \prod_{j=1}^{\ell} \mathbb{F}_{p^D}$ and ciphertext space R_q^2 . Let HE-IOPP be an HE-IOP of proximity. Let $L = \{x_{i,j}\}_{i \in [d], j \in [\ell]}$, $L \subseteq \mathbb{F}_{p^D}$ and let $\text{ct}_1, \dots, \text{ct}_d \in R_q^2$ be alleged ciphertexts such that $\text{HE.Dec}(\text{ct}_i) = (m_{i,1}, \dots, m_{i,\ell}) \in \prod_{j=1}^{\ell} \mathbb{F}_{p^D}$. Finally, let $f \in \mathbb{F}_{p^D}[\mathbf{X}]_{<|L|}$ be the polynomial such that $f(x_{i,j}) = m_{i,j} \in \mathbb{F}_{p^D}$ for every $x_{i,j} \in L$. We say that $\text{ct}_1, \dots, \text{ct}_d \in R_q^2$ define an encrypted polynomial (of f , at L) if there exist admissible circuits such that,*

- On input $\text{ct}_1, \dots, \text{ct}_d \in R_q^2$ and any $(\alpha_1, \dots, \alpha_{\ell}) \in \prod_{j=1}^{\ell} \mathbb{F}_{p^D}$, it returns a ciphertext ct' such that, with overwhelming probability, $\text{HE.Dec}(\text{ct}') = (f(\alpha_1), \dots, f(\alpha_{\ell})) \in \prod_{j=1}^{\ell} \mathbb{F}_{p^D}$.
- On input $\text{ct}_1, \dots, \text{ct}_d \in R_q^2$ to the HE-IOPP, all honestly produced messages within it decrypt correctly (with overwhelming probability).

When we want to make the plaintext polynomial and evaluation domain explicit, we write $(\text{ct}_1, \dots, \text{ct}_d) \in \text{EncPoly}(f, L)$.

In other words, $(\text{ct}_1, \dots, \text{ct}_d) \in \text{EncPoly}(f, L)$ if, given those ciphertexts, it is possible both to compute $\text{EncPoly}(f, \mathbb{F}_{p^D})$ and to show within the HE-IOPP that there exists such an f .

5.1 The HE-Batched-FRI Protocol

The specific HE-IOPP we will employ is the HE transformation (see Definition 13) of the (Batched) FRI protocol. The batched FRI protocol allows a

⁶ It could happen, for a malicious choice of the $\text{ct}_i \in R_q^2$, that $f(a) = \sum_{i=0}^d g_i a^i \forall a \in R_p$ for some $g_i \neq f_i$. In practice, this does not give any power to the adversary: it would be equivalent to putting a wrong polynomial of the right degree within the oracle, which should be caught by the IOP.

prover to prove the δ -correlated agreement of f_1, \dots, f_t by running the FRI protocol on $f = \sum_{i=1}^t \beta_i f_i$ for i.i.d. uniformly random⁷ β_i . In fact, replacing FRI with another IOPP would still result in a δ -correlated agreement test and as we showed before (Theorems 3 and 4), we could use any other IOPP, to which we would previously apply our HE-transformation (Definition 13).

Whereas there are new, concretely more efficient IOPs for circuit satisfiability every year [BCR+19, COS20], a series of variants of the FRI protocol have remained as the most practical choice for an IOPP until this day. Since this is the most stable component of our overall compilers, we provide our HE-Batched-FRI protocol in Fig. 3. We also adapt the results of [BGK+23] concerning round-by-round soundness of FRI to HE-Batched-FRI. Notice that we only need to consider RBR soundness, rather than RBR knowledge soundness, since the former is enough for their δ -correlated hIOP-to-SNARK compiler.

Theorem 5. *Let \mathbb{F} be a finite field, $L_0 \subseteq \mathbb{F}^*$ a smooth multiplicative subgroup of size 2^n , $d_0 = 2^k, \rho = d_0/|L_0| = 2^{k-n}$ and ℓ a positive integer. For any integer $m \geq 3$, $\eta \in (0, \sqrt{\rho}/(2m))$, relative distance $\delta \in (0, 1 - \sqrt{\rho} - \eta)$ and functions $f_1^{(0)}, \dots, f_t^{(0)} : L_0 \rightarrow \mathbb{F}$ for $t \geq 2$ such that at least one of them is δ -far from $RS^{(0)}$, the HE-Batched-FRI protocol (Fig. 3) is complete and has round-by-round soundness error $\epsilon = \max \left\{ \frac{(m+1/2)^7 \cdot |L_0|^2}{3\rho^{3/2}|\mathbb{F}|}, (1-\delta)^\ell \right\}$. Furthermore, under Conjecture 5.12 from [BGK+23], the error can be further reduced to $\epsilon = \max \left\{ \frac{|L_0|^{c_2}}{(\rho\eta)^{c_1}|\mathbb{F}|}, (1-\delta)^\ell \right\}$.*

Proof. Completeness follows from Theorem 2 and Definition 15. RBR soundness follows from Theorem 2 and [BGK+23, Theorem 4.2]. ■

6 Optimisations

To make our construction practical, we introduce a series of optimizations.

On the choice of HE Scheme. Our construction requires an exact HE scheme that supports finite fields as the plaintext space, making it compatible with nearly all modern HE schemes, such as TFHE [CGGI20], BGV [BGV12], and BFV [Bra12, FV12]. CKKS [CKKS17], being approximate, is incompatible with our approach. Thus, our choices are BGV/BFV and TFHE, guided by practical performance and implementation availability, detailed in Sect. 7.

6.1 Tensoring

Recall the HE plaintext space structure from Sect. 2. Ideally, the plaintext ring R_p should split into ℓ copies of \mathbb{F}_{p^D} , where D meets FRI security requirements, i.e., $|\mathbb{F}_{p^D}| \approx 2^{256}$. This requires \mathbb{F}_p to contain roots of unity of order at most

⁷ We use i.i.d uniformly random coefficients, rather than powers of a single β , since otherwise we would incur an $O(n)$ soundness loss, see [BCI+20, BGK+23].

The HE-Batched-FRI protocol

Setup: Agree on the following:

- A HE scheme with ciphertext space R_q^2 and plaintext space $R_p \cong \prod_{j=1}^{\ell} \mathbb{F}_{p^D}, p \neq 2$, satisfying $2^n | (p^D - 1)$ for some positive integer n .
- A multiplicative group $L_0 = \{\omega, \dots, \omega^{2^n}\} \subseteq \mathbb{F}_{p^D}^*$ of order 2^n , i.e. $L_0 = \langle \omega \rangle$.
- The rate of the Reed-Solomon code, $\rho = 2^{-R}$ for a positive integer R .
- A number of rounds $r < n - R$, each of which will use a domain $L_{i+1} = \{x^2 \mid x \in L_i\}$, i.e. $L_{i+1} = \langle \omega^{2^i} \rangle$.

Input: For $i = 1, \dots, t$, alleged ciphertexts $(c_{i,1}, \dots, c_{i,2^n/\ell}) \in R_q^2$ which allegedly satisfy $(c_{i,1}, \dots, c_{i,2^n/\ell}) \in \text{EncPoly}(f_i, L_0)$ for some $f_i(\mathbf{X}) \in \mathbb{F}_{p^D}[\mathbf{X}]_{<2^n-R}$. In other words, it should be that $f_i|_{L_0} \in \text{RS}[\mathbb{F}_{p^D}, L_0, 2^{n-R}]$. \mathcal{V} has oracles $\{\{\text{EncPoly}(f_i, L_0)\}\}_{i=1}^t$, whereas \mathcal{P} has the underlying ciphertexts $\{\text{EncPoly}(f_i, L_0)\}_{i=1}^t$.

Commit phase: \mathcal{P} batches the encryptions of functions f_1, \dots, f_t into a single encryption of a function f . Afterwards, it sequentially constructs a series of oracles to “foldings” of encryptions of that function.

1. Obtain challenges $\beta_1, \dots, \beta_t \stackrel{\$}{\leftarrow} \mathbb{F}_{p^D}$ from the verifier. Implicitly define the oracle $\llbracket f|_{L_0} \rrbracket_{\text{HE}} = \sum_{i=1}^t \beta_i \cdot \llbracket f_i|_{L_0} \rrbracket_{\text{HE}}$ from the oracles to the encrypted polynomials $\{\text{EncPoly}(f_i, L_0)\}_{i=1}^t$.
2. For $0 \leq i < r$:
 - Obtain a challenge $\alpha_i \stackrel{\$}{\leftarrow} \mathbb{F}_{p^D}$ from the verifier.
 - \mathcal{P} computes and sends the oracle $\llbracket \text{EncPoly}(f^{(i+1)}, L_{i+1}) \rrbracket$, where $L_{i+1} = \langle \omega^{2^{i+1}} \rangle$ and $f^{(i+1)} : L_{i+1} \rightarrow \mathbb{F}_{p^D}$ is allegedly such that (for $j \in [2^{n-i}]$):

$$f^{(i+1)}(\omega^{2^{i+1}j}) = \frac{f^{(i)}(\omega^{2^i j}) + f^{(i)}(-\omega^{2^i j})}{2} + \alpha_i \cdot \frac{f^{(i)}(\omega^{2^i j}) - f^{(i)}(-\omega^{2^i j})}{2 \cdot \omega^{2^i j}}.$$

3. The last oracle (to an encryption of) $f^{(r)} : L_r \rightarrow \mathbb{F}_{p^D}$, $L_r = \langle \omega^{2^{r-1}} \rangle$ is allegedly an encryption of a polynomial of degree strictly less than $\rho \cdot |L_r|$, so \mathcal{P} can just directly send it as $\text{EncPoly}(f^{(r)}, L_r)$ to \mathcal{V} .

Query phase: \mathcal{V} , using theHE secret key sk in order to decrypt the encrypted polynomials within the oracles, checks the consistency of the messages sent by the prover. If any of the checks below fails, reject and abort; otherwise accept.

1. For $0 \leq \ell < m$, \mathcal{V} does the following in parallel:
 - Sample a random $\mu_\ell \stackrel{\$}{\leftarrow} L_0$.
 - For $0 \leq i < r$, using the oracles $\{\llbracket \text{EncPoly}(f^{(j)}, L_j) \rrbracket\}_{j=0}^r$ and challenges $\{\alpha_j\}_{j=0}^{r-1}$ from the commit phase, check whether:

$$f^{(i+1)}(\mu_\ell^{2^{i+1}}) \stackrel{?}{=} \frac{f^{(i)}(\mu_\ell^{2^i}) + f^{(i)}(-\mu_\ell^{2^i})}{2} + \alpha_i \cdot \frac{f^{(i)}(\mu_\ell^{2^i}) - f^{(i)}(-\mu_\ell^{2^i})}{2 \cdot \mu_\ell^{2^i}}.$$

Notice that the verifier needs to decrypt the values $f^{(i+1)}(\mu_\ell^{2^{i+1}})$, $f^{(i)}(\mu_\ell^{2^i})$ and $f^{(i)}(-\mu_\ell^{2^i})$ inside the oracles in order to do this.

2. Decrypt $\text{EncPoly}(f^{(r)}, L_r)$ and check whether $f^{(r)}|_{L_r} \stackrel{?}{\in} \text{RS}[\mathbb{F}_{p^D}, L_r, d]$.

Fig. 3. The HE-Batched-FRI protocol.

$2N/D$, where N is the cyclotomic ring degree. Given the HE scheme's requirement that $\log(N) \in 11, \dots, 17$, p would have to be smaller than $2N/D$. Additionally, FRI requires \mathbb{F}_{p^D} to have a 2^n -th root of unity, thus $pD > 2^n$. In practice, using roots of unity in \mathbb{F}_p (requiring $p > 2^n$) allows the NTT (Sect. 6.2) to run D times faster.

Combining all requirements (FRI, HE, implementation), $2^n = d\rho^{-1} < p < 2N/D$, restricting input polynomial size to $d < 2N\rho/D$. This is feasible with parameter sets like $(\log(N), \rho, D) = (17, 1/2, 6)$, allowing d up to 2^{14} . However, better-performing parameters, such as $(\log(N), \rho, D) = (14, 1/16, 12)$, restrict d to around 2^6 , almost entirely restricting the use of FRI use.

To address this, we use a field extension \mathbb{F}_{p^D} of \mathbb{F}_p in our HE-FRI protocol. When evaluating an encrypted polynomial on an element of \mathbb{F}_{p^D} , we emulate the arithmetic of \mathbb{F}_{p^D} through a circuit. This results in D HE ciphertexts, which together encrypt a single value in \mathbb{F}_{p^D} . Alternatively, an intermediate approach uses a value d' , with the plaintext space as $\mathbb{F}_{p^{d'}}$, and emulates \mathbb{F}_{p^D} arithmetic with D/d' ciphertexts. The ideal value of d' depends on the application, since increasing the plaintext modulus increases all HE parameters.

6.2 Shallow Reed-Solomon Encoding

Reed-Solomon encoding consists of interpreting data as a polynomial of degree $(d-1)$ and evaluating it at $2^n = d\rho^{-1}$ independent points. Polynomial evaluation is linear and simply evaluating it 2^n times would result in quadratic performance. The Fast Fourier Transform is a staple solution for this problem, enabling the evaluation in $O(2^n \log 2^n)$ operations. Commonly, however, it is implemented as a circuit with depth $\log 2^n$, which poses some challenges for its homomorphic evaluation. Fortunately, FFTs, and, more specifically, Number-Theoretic Transforms (NTTs), their generalization to finite fields, are ubiquitous in the FHE literature, and solutions for evaluating them with small depth are very well established [CG99, GPvL23]. In this work, we adopt a radix- k NTT of parametrizable depth for some $k \in [2, \sqrt{2^n}]$ optimized based on practical performance.

The NTT can be especially costly when running batched FRI, as the RS codeword needs to be calculated for each polynomial individually. This cost can be minimized by exploiting the HE scheme packing to perform the NTT over several polynomials at once. We consider the following strategies for packing.

- **Single polynomial packing:** Let k be a single polynomial $k = \sum_{i=0}^{d-1} k_i X^i \in \mathbb{F}_p[X]$. We encrypt k in an array of d/N ciphertexts ct , such that ct_i encrypts $\sum_{j=0}^{d-1} k_{i \cdot N + j} X^j$. The main advantage of this approach is the significantly reduced memory usage, as we process one polynomial at a time. Conversely, evaluating the NTT algorithm with this packing requires performing permutations within each ciphertext [CG99], which is an expensive process compared to the other operations needed for evaluating the NTT.
- **Batched polynomial packing:** Let k be an N -sized list of polynomials with maximum degree $(d-1)$, and $k_{i,j}$ be the coefficient of degree j of the i -th polynomial. We encrypt k in an array of d ciphertexts ct_i , such that the j -th

slot of ct_i encrypts $k_{j,i}$. The main advantage of this approach is avoiding the aforementioned permutations, as each coefficient of a polynomial would be in different ciphertexts. For large polynomials, the memory requirements to run the NTT with this packing might be impractical, however.

In both cases, even though FRI is defined over \mathbb{F}_{p^D} , we perform the entire NTT in \mathbb{F}_p by selecting roots of unity in \mathbb{F}_p , as roots of unity in \mathbb{F}_{p^D} would bring negligible advantage compared to the size of p (as discussed in Sect. 6.1). Further, the goal of the NTT is to create a redundant representation of the polynomial (*i.e.*, the RS codeword), which consumes more memory to be stored. In this way, storing the codewords could represent a problem, even if storing the original polynomials was not. This is the main aspect we consider when choosing which type of packing we adopt. Mixed packing approaches could likely be a better solution for this problem, but developing them is not within our scope.

6.3 Shallow Folding

In the commit phase of HE-FRI (Fig. 3), \mathcal{P} needs to compute a series of oracles $\llbracket \text{EncPoly}(f^{(i)}, L_i) \rrbracket$. Let us denote $f^{(i+1)} = \text{Fold}(f^{(i)}, \alpha_i)$. The complexity of producing all such foldings as described there is $O(2^n)$, while the depth⁸ is n . To reduce depth, we replace the FFT-like algorithm with a DFT-like algorithm. We compute the first layer of the `Fold` operation as usual, then pre-compute constants for the following layers, expressing each as a composition `Fold` \circ \dots \circ `Fold`. Each layer can now be expressed as inner products of the original polynomial, reducing the depth to 1, while increasing the complexity to $O(2^n \log(2^n))$. This does not affect the overall FRI complexity (dominated by the NTT). The Verifier's side remains unchanged. The full algorithm is presented in the full version [ACGS23].

6.4 Fast Decryption

In RLWE-based cryptography, decrypting small amounts of data may incur a significant overhead depending on the adopted parameters. An RLWE sample of dimension N encrypts up to N messages in \mathbb{F}_p , which can all be decrypted at once with cost $O(N \log N)$. However, if one wants to decrypt just a single message in \mathbb{F}_p , the cost would be at least $O(N)$, which represents a performance overhead of $N/\log(N)$ times compared to the amortized cost of decrypting all messages at once. During the commit phase of HE-FRI, the prover performs computation using RLWE samples of dimension N encrypting N messages in \mathbb{F}_p . During the query phase, however, the verifier only needs to learn two evaluation points in \mathbb{F}_{p^D} per round for each linearity check. In this way, if the prover provides these points packed in a ciphertext of dimension N , it would impose a performance overhead of at least $N/(2D)$ times for the verifier compared to an optimal RLWE decryption (*i.e.*, it would be decrypting at least $N/(2D)$

⁸ n is the logarithm of the codeword size. Folding is a linear algorithm.

more messages than necessary). To improve on this, we propose three possible approaches. We summarise them here and present the details in the full version [ACGS23].

Repacking. The simplest way of minimizing decryption costs is to reduce the ciphertext dimension. This can be achieved in several ways. In this work, since the prover also needs to arithmetically manipulate points individually, we choose to extract the points to LWE samples and repack them using a key-switching algorithm. We pack the two points needed per linearity check in the same RLWE samples, further minimizing decryption costs.

Decomposition and Recomposing. While the repacking already enables us to significantly reduce the ciphertext dimension, we are still limited by the value of p . Specifically, the ciphertext dimension depends on the ciphertext modulus for security, which, in turn, depends on the plaintext modulus p . To enable further reductions in the size of N , we introduce a decomposition and recomposing procedure based on techniques introduced in [CLOT21] and described in [CGGI20]. Our proposal starts from the observation that since the verifier does not compute any homomorphic operations on these samples (it only decrypts them), *we do not need to preserve any homomorphic properties*. In fact, once the commit phase is finished, the evaluation points can be treated simply as strings of bits, and the goal becomes to encrypt them in the smallest possible RLWE ciphertext. Considering this, we homomorphically decompose elements in \mathbb{F}_p in digits of size $\log \bar{p}$ for some $\bar{p} < p$. This process enables us to repack the evaluation points in RLWE ciphertexts with smaller moduli and, hence, smaller dimensions. We present our full recomposition process in the full version [ACGS23].

Adding Samples. Adding samples appears as an alternative to the decomposition. The repacking procedure mentioned above allows us to reduce the dimension of the RLWE samples that are sent to the verifier, but we are still not using all the coefficients in the sample. Concretely, our implementation uses ciphertexts with $N = 512$ to only encrypt $2D = 32$ messages. Considering this, the verifier can further minimize decryption costs by adding rotations of ciphertexts together and decrypting messages from multiple ciphertexts at once. We note that rotation can be done considering *coefficient representation*, which is inexpensive. While this efficiently minimises the number of decryptions, it does not, contrary to the decomposition, reduce the cost of hashing operations or proof size.

7 Experimental Results

Operating over the plaintext space allows compatibility with nearly all HE schemes (except CKKS), enabling various implementation methods. For this proof of concept, we focus on a simple and efficient implementation that demonstrates practical execution times for both the prover and verifier. While we make specific choices of schemes and techniques, our construction remains compatible

with most existing schemes. the full version [ACGS23]. provides a performance characterization based on the number of basic operations executed by each party, with and without the optimizations in Sect. 6. This broader view of HE-FRI performance helps extrapolate the results to other parameters or schemes.

7.1 Practical Parameters

For the security of FRI, we consider parameters established by previous literature [BGK+23], reproduced in Table 1. There are also practical parameters that are required for functionality, as we discussed in Sect. 6.1. Considering this, Table 2 presents the main choices of parameters according to the maximum size of the input polynomial that they support. We note that, for every parameter, the maximum size of the input polynomial can be increased by up to D times at the cost of D times more expensive NTT (as also discussed in Sect. 6.1).

Table 1. Security parameters we adopt for FRI, based on estimates of [BGK+23] using Conjecture 5.12 from [BGK+23] We approximate the size of the field for practical reasons.

Parameter	$\log_2(\mathbb{F}_{p^D})$	ρ	m	δ
FRI ₀	251-269	1/2	102	0.5
FRI ₁		1/4	51	0.75
FRI ₂		1/8	34	0.875
FRI ₃		1/16	26	0.937

Table 2. Practical parameters for FRI based on the maximum size of the input polynomial d .

Maximum input size $\log_2(d)$	D	p	$\log_2(p)$	$\log_2(\mathbb{F}_{p^D})$
15	16	65537	16.0	256.0
20	11	23068673	24.5	269.1
25	9	469762049	28.8	259.3
30	7	75161927681	36.1	252.9
35	7	206158430209	37.6	263.1
40	6	6597069766657	42.6	255.5
45	5	1337006139375617	50.2	251.2

7.2 Proof-of-Concept Implementation

We build our proof of concept over the implementation of [S+21], a Python-implemented version of FRI for prime fields. We extend it to work over the extension field \mathbb{F}_{p^D} and connect it to optimized FHE libraries to implement the encrypted arithmetic. We consider polynomials of degree up to 2^{15} and present results for parameter sets FRI_0 and FRI_3 , which are optimized for the verifier and the prover, respectively. We run all the experiments in a `c6i.metal` instance (Intel Xeon 8375C at 3.5 GHz with 256 GiB of RAM) on AWS. The prover is parallelized to use up to 32 threads and the verifier is single-threaded.

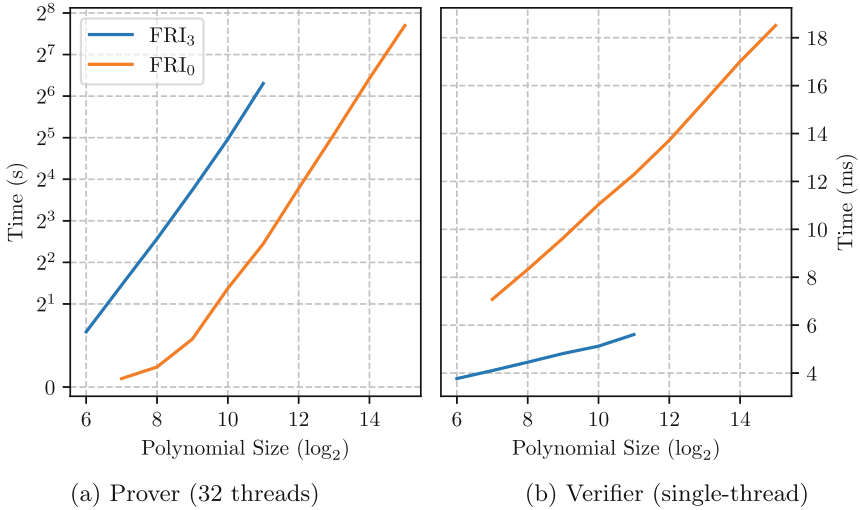


Fig. 4. Performance of HE-FRI using parameter sets FRI_3 and FRI_0 for $D = 16$ for 4096 polynomials (batched). Detailed results are provided in the full version [ACGS23].

Prover Performance. Figure 4a shows the results for the prover. It includes the execution time of the RS encoding, batching, and folding procedures. For the RS encoding, we implemented a generic RNS-based homomorphic NTT implementation built over Intel HEXL [BKS+21]. Its performance should be similar to most commonly used HE libraries, as HEXL is used for RNS implementations on libraries such as HELib [HS20] and OpenFHE [ABBB+22]. We use depth-2 NTTs with parallelized recursive calls (up to a maximum of 32 threads) and perform batched polynomial packing (Sect. 6.2) to compute it on $N = 4096$ polynomials at once. Our construction is flexible to the HE encoding, as we may later move to the required encodings during batching, as the full version [ACGS23] details.

Our folding procedure, on the other hand, has fixed depth (Sect. 6.3) and is the last procedure to run before decryption. As such, it requires a much smaller

ciphertext modulus and can be evaluated using single-precision (non-RNS) implementations. In this way, whenever possible, we evaluate it using implementation techniques from TFHE [CGGI20] using the MOSFHET library [GBA24]. Despite being asymptotically quasi-linear, we note that the execution time increases almost linearly with the polynomial size, which is a result of better parallel efficiency and the overhead introduced by repacking (which is linear). With FRI_0 , it takes less than 0.5 s to run for polynomials of degree bound up to 2^8 , going up to 207 s for polynomials of degree bound 2^{15} .

Verifier Performance. Figure 4b shows the verifier results. Verification in FRI is sublinear and typically runs in milliseconds. Hence, Python is not a good fit to showcase practical timings, and we implemented an optimized version of it in C. We only present a single-threaded version, but we note that it is trivially parallelizable and could be significantly accelerated for larger parameters. We show results for FRI_0 and FRI_3 , noting that other parameters can also affect verifier performance, particularly decryption costs. To minimize these, we only considered repacking without decompositions, but the full version [ACGS23] discusses further improvements. Our current setup allows for batched verification of 4096 codewords in as low as 6ms, with performance scaling linearly with the batch size. Future work includes optimizing for smaller batches and other parameters.

Acknowledgements. We would like to thank Zvika Brakerski for comments about our repacking optimization for the HE-Batched-FRI protocol. We also want to thank Alexander R. Block, Albert Garreta, Jonathan Katz, Justin Thaler, Pratyush Ranjan Tiwari and Michał Zajac for a useful conversation about their work [BGK+23] and confirming that their analysis does not require finite fields to be prime.

This work was partly done while A. Guimarães was a Ph.D. student at University of Campinas, Brazil. He was supported by the São Paulo Research Foundation under grants 2013/08293-7, 2019/12783-6, and 2021/09849-5. This work is partially funded by the European Union (GA 101096435). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Commission. Neither the European Union nor the European Commission can be held responsible for them.

References

- ABBB+22. Ahmad Al Badawi, Jack Bates, Flavio Bergamaschi, David Bruce Cousins, Saroja Erabelli, Nicholas Genise, Shai Halevi, Hamish Hunt, Andrey Kim, Yongwoo Lee, Zeyu Liu, Daniele Micciancio, Ian Quah, Yuriy Polyakov, Saraswathy R.V., Kurt Rohloff, Jonathan Saylor, Dmitriy Sponitsky, Matthew Triplett, Vinod Vaikuntanathan, and Vincent Zucca. OpenFHE: Open-Source Fully Homomorphic Encryption Library. In *Proceedings of the 10th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, WAHC'22, page 53-63, New York, NY, USA, 2022. Association for Computing Machinery.

- ACGS23. Diego F. Aranha, Anamaria Costache, Antonio Guimarães, and Eduardo Soria-Vazquez. HELIOPOLIS: Verifiable computation over homomorphically encrypted data from interactive oracle proofs is practical. *Cryptology ePrint Archive*, Paper 2023/1949, 2023.
- ACY23. Gal Arnon, Alessandro Chiesa, and Eylon Yogev. IOPs with inverse polynomial soundness error. *Cryptology ePrint Archive*, 2023.
- AHH+24. Nuttapon Attrapadung, Goichiro Hanaoaka, Ryo Hiromasa, Yoshihiro Koseki, Takahiro Matsuda, Yutaro Nishida, Yusuke Sakai, Jacob C. N. Schuldt, and Satoshi Yasuda. Privacy-preserving verifiable CNNs. In Christina Pöpper and Lejla Batina, editors, *Applied Cryptography and Network Security*, pages 373–402, Cham, 2024. Springer Nature Switzerland.
- APS15. Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.
- BBHR18. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Fast reed-solomon interactive oracle proofs of proximity. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *ICALP 2018*, volume 107 of *LIPICs*, pages 14:1–14:17. Schloss Dagstuhl, July 2018.
- BCCW19. Fabian Boemer, Anamaria Costache, Rosario Cammarota, and Casimir Wierzynski. nGraph-HE2: A high-throughput framework for neural network inference on encrypted data. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 45–56, 2019.
- BCFK21. Alexandre Bois, Ignacio Cascudo, Dario Fiore, and Dongwoo Kim. Flexible and efficient verifiable computation on encrypted data. In Juan Garay, editor, *PKC 2021, Part II*, volume 12711 of *LNCS*, pages 528–558. Springer, Heidelberg, May 2021.
- BCI+20. Eli Ben-Sasson, Dan Carmon, Yuval Ishai, Swastik Kopparty, and Shubhangi Saraf. Proximity gaps for reed-solomon codes. In *61st FOCS*, pages 900–909. IEEE Computer Society Press, November 2020.
- BCR+19. Eli Ben-Sasson, Alessandro Chiesa, Michael Riabzev, Nicholas Spooner, Madars Virza, and Nicholas P. Ward. Aurora: Transparent succinct arguments for R1CS. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 103–128. Springer, Heidelberg, May 2019.
- BCS16. Eli Ben-Sasson, Alessandro Chiesa, and Nicholas Spooner. Interactive oracle proofs. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 31–60. Springer, Heidelberg, October / November 2016.
- BGBE19. Alon Brutzkus, Ran Gilad-Bachrach, and Oren Elisha. Low latency privacy preserving inference. In *International Conference on Machine Learning*, pages 812–821. PMLR, 2019.
- BGK+23. Alexander R. Block, Albert Garreta, Jonathan Katz, Justin Thaler, Pratyush Ranjan Tiwari, and Michał Zając. Fiat-Shamir Security of FRI and Related SNARKs. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, pages 3–40, Singapore, 2023. Springer Nature Singapore.
- BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012*, pages 309–325. ACM, January 2012.

- BKS+21. Fabian Boemer, Sejun Kim, Gelila Seifu, Fillipe D.M. de Souza, and Vinodh Gopal. Intel HEXL: Accelerating Homomorphic Encryption with Intel AVX512-IFMA52. In *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, WAHC '21, page 57–62, New York, NY, USA, 2021. Association for Computing Machinery.
- BMMP18. Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Pailier. Fast homomorphic evaluation of deep discretized neural networks. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 483–512. Springer, Heidelberg, August 2018.
- Bra12. Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 868–886. Springer, Heidelberg, August 2012.
- BV14. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. *SIAM Journal on computing*, 43(2):831–871, 2014.
- CCH+19. Ran Canetti, Yilei Chen, Justin Holmgren, Alex Lombardi, Guy N. Rothblum, Ron D. Rothblum, and Daniel Wichs. Fiat-Shamir: from practice to theory. In Moses Charikar and Edith Cohen, editors, *51st ACM STOC*, pages 1082–1090. ACM Press, June 2019.
- CCL15. Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 3–22. Springer, Heidelberg, August 2015.
- CG99. Eleanor Chu and Alan George. *Inside the FFT Black Box: Serial and Parallel Fast Fourier Transform Algorithms*. CRC Press, November 1999. Google-Books-ID: 30S3kRiX4xgC.
- CGGI20. Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. TFHE: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, 33(1):34–91, January 2020.
- CKKS17. Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. Homomorphic encryption for arithmetic of approximate numbers. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 409–437. Springer, Heidelberg, December 2017.
- CLOT21. Ilaria Chillotti, Damien Ligier, Jean-Baptiste Orfila, and Samuel Tap. Improved programmable bootstrapping with larger precision and efficient arithmetic circuits for TFHE. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part III*, volume 13092 of *LNCS*, pages 670–699. Springer, Heidelberg, December 2021.
- CMS19. Alessandro Chiesa, Peter Manohar, and Nicholas Spooner. Succinct arguments in the quantum random oracle model. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 1–29. Springer, Heidelberg, December 2019.
- COS20. Alessandro Chiesa, Dev Ojha, and Nicholas Spooner. Fractal: Post-quantum and transparent recursive proofs from holography. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 769–793. Springer, Heidelberg, May 2020.

- CP19. Benjamin R Curtis and Rachel Player. On the feasibility and impact of standardising sparse-secret LWE parameter sets for homomorphic encryption. In *Proceedings of the 7th ACM Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 1–10, 2019.
- DDGR20. Dana Dachman-Soled, Léo Ducas, Huijing Gong, and Mélissa Rossi. LWE with side information: Attacks and concrete security estimation. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 329–358. Springer, Heidelberg, August 2020.
- DPSZ12. Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multi-party computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Heidelberg, August 2012.
- FGP14. Dario Fiore, Rosario Gennaro, and Valerio Pastro. Efficiently verifiable computation on encrypted data. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 844–855. ACM Press, November 2014.
- FNP20. Dario Fiore, Anca Nitulescu, and David Pointcheval. Boosting verifiable computation on encrypted data. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part II*, volume 12111 of *LNCS*, pages 124–154. Springer, Heidelberg, May 2020.
- FB12. Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Report 2012/144, 2012. <https://eprint.iacr.org/2012/144>.
- GBA24. Antonio Guimarães, Edson Borin, and Diego F. Aranha. MOSFET: Optimized Software for FHE over the Torus. *Journal of Cryptographic Engineering*, July 2024.
- Gen09a. Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.
- Gen09b. Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st ACM STOC*, pages 169–178. ACM Press, May / June 2009.
- GGP10. Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 465–482. Springer, Heidelberg, August 2010.
- GGW23. Sanjam Garg, Aarushi Goel, and Mingyuan Wang. How to prove statements obliviously? Cryptology ePrint Archive, Paper 2023/1609, 2023. <https://eprint.iacr.org/2023/1609>.
- GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.
- GNS23. Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. Rinocchio: SNARKs for ring arithmetic. *Journal of Cryptology*, 36(4):41, October 2023.
- GPvL23. Antonio Guimarães, Hilder V. L. Pereira, and Barry van Leeuwen. Amortized bootstrapping revisited: Simpler, asymptotically-faster, implemented. In Jian Guo and Ron Steinfeld, editors, *Advances in Cryptology – ASIACRYPT 2023*, pages 3–35, Singapore, 2023. Springer Nature Singapore.

- GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Daniel Wichs. Leveled fully homomorphic signatures from standard lattices. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 469–477. ACM Press, June 2015.
- Hab22. Ulrich Haböck. A summary on the fri low degree test. Cryptology ePrint Archive, Paper 2022/1216, 2022. <https://eprint.iacr.org/2022/1216>.
- HS20. Shai Halevi and Victor Shoup. Design and implementation of HELib: a homomorphic encryption library. Cryptology ePrint Archive, Report 2020/1481, 2020. <https://eprint.iacr.org/2020/1481>.
- S+21. Szepieniec et al. Anatomy of a stark - tutorial for starks with supporting code in python, November 2021. <https://github.com/aszepieniec/stark-anatomy>
- SV14. Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *DCC*, 71(1):57–81, 2014.

Zero-knowledge Protocols



Interactive Line-Point Zero-Knowledge with Sublinear Communication and Linear Computation

Fuchun Lin^(✉), Chaoping Xing, and Yizhou Yao

Shanghai Jiao Tong University, Shanghai, China
{`linfuchun, xingcp, yaoyizhou0620`}@sjtu.edu.cn

Abstract. Studies of vector oblivious linear evaluation (VOLE)-based zero-knowledge (ZK) protocols flourish in recent years. Such ZK protocols feature optimal prover computation and a flexibility for handling arithmetic circuits over arbitrary fields. However, most of them have linear communication, which constitutes a bottleneck for handling large statements in a slow network. The pioneer work AntMan (CCS'22), achieved sublinear communication for the first time within VOLE-based ZK, but lost the advantage of fast proving. In this work, we propose two new VOLE-based ZK constructions that achieve sublinear communication and linear computation, simultaneously. Let \mathcal{C} be a circuit with size S , input size n , and depth d . In particular, our first ZK, specialized for layered circuits, has communication $O(n + d \log S)$, while our second ZK can be used to prove general circuits and has communication $O(n + d \log S + d^2)$. Our results are obtained by introducing the powerful sum-check techniques from the mature line of works on interactive proofs into the context of VOLE-based ZK for the first time. Reminiscent of the *non-interactive* line-point zero-knowledge proof system (ITC'21), we introduce an *interactive line-point zero-knowledge* (ILPZK) proof system, which closely connects with VOLE-based ZK protocols. In addition, our works also enrich the studies of ZK based on interactive proofs, with new interesting features (e.g., having information-theoretic UC-security, naturally supporting any field) achieved.

1 Introduction

A proof system allows a prover to convince a verifier that a given input x belongs to some language \mathcal{L} . In the literature, *circuit satisfiability* is a popular **NP** language, where the prover proves to the verifier that a given circuit $\mathcal{C} : \mathbb{F}^n \rightarrow \mathbb{F}^t$ is satisfiable (i.e., there exists some witness $\mathbf{w} \in \mathbb{F}^n$ such that $\mathcal{C}(\mathbf{w}) = \mathbf{1}$). Furthermore, a zero-knowledge (ZK) proof of knowledge system guarantees a valid proof can only be generated by the one who holds the witness \mathbf{w} , and it reveals nothing about \mathbf{w} beyond $\mathcal{C}(\mathbf{w}) = \mathbf{1}$.

Yizhou Yao is the first author as well as the main contributor of this paper.

© International Association for Cryptologic Research 2025
K.-M. Chung and Y. Sasaki (Eds.): ASIACRYPT 2024, LNCS 15488, pp. 337–366, 2025.
https://doi.org/10.1007/978-981-96-0935-2_11

Though studies of ZK proof systems date back to 1980s [20], they have not been brought into real-life applications until past few years ago. There are two mainstream focuses on improving the concrete efficiency of ZK proofs, prover time, and proof size (which constitutes a bottleneck of verification time). For prover time, the best one can hope for is that generating a proof is as fast as verifying the witness in the clear. For proof size, proofs with proof size as small as possible (e.g., constant) are preferred, so that fast verification for large statements becomes possible. However, achieving both goals simultaneously is generally very challenging, in the sense that compressing the proof size is usually accompanied by consumption of prover’s computation resources. To our best knowledge, there are a few existing ZK proof systems that achieve linear prover time¹ and sublinear (in the circuit size) proof size simultaneously. We give a brief overview on them as follows.

ZK with Linear Prover Time and Sublinear Proof Size. The first such proof system falls within the scope of *interactive oracle proofs* (IOP), with a line of works [8, 9, 21, 35]. Brakedown [21], as the state-of-the-art, exploits a tensor structure of linear combinations, and makes use of linear-time encodable codes. Due to the asymptotic nature of known constructions for linear-time encodable codes [18, 21], the prover’s computational overhead might be relatively large for small or medium-sized statements. Moreover, these works operate over large fields with field size at least $\Omega(|C|)$, restricting the application scenarios.

The second approach follows the GKR interactive proof (IP) protocol [19], with a line of works [31, 34, 37, 38]. At a high level, these works start with optimizations of GKR that have linear prover time, and incorporate a commitment scheme to achieve zero-knowledge. The prover computation in original GKR is around cubic in the circuit size, dominated by evaluating multi-variate polynomials at multiple points. The computational overhead was finally optimized to constant due to efforts in a series of works [14, 29, 30, 34], through exploiting that these polynomials and points are highly structured. On a separate note, original GKR assumes a layered circuit, in which each gate takes input only from gates in the previous layer. The recent breakthrough [37] extends GKR to general circuits and maintains a linear time prover, by carefully describing the much more complicated relations among layers. We finally summarize the different commitment schemes used in these works². In Hyrax [31], the underlying commitment is Pedersen commitment [26], while in Libra [34] and Virgo++ [37], the authors use a pairing-based polynomial commitment. Therefore, all these three protocols are not post-quantum. Moreover, the use of a pairing-based polynomial commitment not only leads to an inherent trusted setup, but also incurs a bigger computational overhead when the witness size is close to the circuit size.

VOLE-based Zero-Knowledge Proofs. Very recently, a line of works [10–12] studying how to efficiently generate pseudorandom correlations between par-

¹ We say a ZK proof has linear prover time, if the prover’s computation is only a constant times larger than that of verifying the witness in the clear.

² We remark that Virgo [38] uses a polynomial commitment based on FRI [6], leading to prover computation $\mathcal{O}(|C| + n \log n)$, where n is the input size.

ties gave birth to ZK proofs based on random *vector oblivious linear evaluation* (VOLE) correlations. In general, VOLE-based ZK protocols [2, 5, 16, 17, 24, 32, 36] have fascinating performance on the prover side, where the prover only pays a very small constant computational overhead, and proving can be done in a streaming fashion to reduce memory cost. One typical example is the *line-point zero-knowledge* (LPZK) [17], with only $4\times$ to $7\times$ prover computational overhead. LPZK requires sending one field element per multiplication gate and its followup work [16] reduced the communication complexity to $1/2$ field element per multiplication gate for layered circuits, which is still linear in the circuit size³. Another optimization of LPZK proposed the QuickSilver [36] demonstrating an important advantage of VOLE-based ZK protocols compared against other proof systems. That is VOLE-based ZK protocols can be easily adapted to prove arithmetic circuits over small fields (even Boolean circuits) with online communication independent of the security parameter. We refer to a recent survey [4] for more details of the VOLE-based ZK literature.

Most VOLE-based ZK protocols are not succinct, with linear proof size and verifier costs almost the same as the prover. The only one exception is AntMan [33], which achieves sublinear proof size but at the cost of increasing prover time to quasi-linear and relying on an additively homomorphic encryption (AHE) scheme. To our best knowledge, there is no VOLE-based ZK protocol achieving linear prover time and sublinear proof size simultaneously. As previous ZK proofs with such properties are realized under frameworks of succinct proofs (either requires a trusted setup, or assumes strong cryptographic assumptions, or only supports large fields), we ask the following question:

Can we realize ZK protocols with linear prover time and sublinear proof size in the context of VOLE-based ZK?

1.1 Our Contributions

We bring the powerful sum-check protocol [25] along with multi-linear extensions (MLE) into the study of VOLE-based ZK protocols. This is the first time a non-trivial (compressing) classical proof technique is used in this new paradigm. To distinguish from the conventional VOLE-based ZK protocols, we formulate our protocols under a new framework that we dub *interactive line-point zero-knowledge (ILPZK)*. In an LPZK proof, the prover \mathcal{P} *independently* generates an affine line $\mathbf{v}(x) := \mathbf{a} \cdot x + \mathbf{b}$ in an ℓ -dimensional vector space \mathbb{F}^ℓ from the circuit \mathcal{C} and the witness \mathbf{w} . Then the verifier \mathcal{V} queries a single point $\mathbf{v}(\Delta) := \mathbf{a} \cdot \Delta + \mathbf{b}$ on this line, which allows \mathcal{V} to check the correctness of the computation of every gate. This “gate-by-gate” nature leads to a barrier of squashing the proof length (i.e., the dimension ℓ) to sublinear.

In our protocols, due to the introduction of interactive sum-check, the affine line $\mathbf{v}(x) := \mathbf{a} \cdot x + \mathbf{b}$ is generated *collectively* by the prover \mathcal{P} and the verifier \mathcal{V} , where \mathcal{V} 's participation is in the form of providing random challenges in each

³ For special type of statements, sublinear proof size constructions were reported, for example, conjunctions [5] and low-degree polynomials [36].

round of sum-check. The involvement of \mathcal{V} in the generation of the affine line makes it possible for implementing probabilistic checking within $\mathbf{v}(x)$, in the sense that wire values are “compressed” by MLE encodings. This is the reason why the proof size can be made sublinear in the circuit size and the verifier can be light-weight. We believe that ILPZK (see Definition 2 for a formal description) captures the essence of our new protocols and may have independent interest in its own right.

ILPZK for Layered Arithmetic Circuits. Our ILPZK proof for satisfiability of layered arithmetic circuits achieves linear prover time and strict sublinear proof size $\mathcal{O}(n + d \log S)$, where n is the witness length and S is the circuit size.

Theorem 1 (ILPZK for layered arithmetic circuit satisfiability). *Let $\mathcal{C} : \mathbb{F}^n \rightarrow \mathbb{F}^{n'}$ be a layered (log-space uniform) arithmetic verification circuit with depth d and number of gates S . There exists an ILPZK proof that proves the satisfiability of \mathcal{C} with the following features:*

- *The prover runs in time $\mathcal{O}(S)$.*
- *The verifier runs in time $\mathcal{O}(n + n' + d \log S + T)$, where $\mathcal{O}(T)$ is the time for evaluating MLEs, and is sublinear for log-space uniform circuits.*
- *Round complexity is $\mathcal{O}(d \log S)$.*
- *Proof length is $\mathcal{O}(n + d \log S)$.*
- *Soundness error is $\mathcal{O}(\frac{d \log S}{|\mathbb{F}|})$.*

ILPZK for generic arithmetic circuits. Our ILPZK construction for general arithmetic circuits achieves linear prover time as well and mostly has sublinear proof size except some extreme cases. While each gate of a generic circuit may take inputs from all previous layers, one can still separate a generic circuit into d “layers” where each gate takes at least one input from the previous layer, and d also refers to the depth of the circuit.

Theorem 2 (ILPZK for Generic Arithmetic Circuit Satisfiability). *Let $\mathcal{C} : \mathbb{F}^n \rightarrow \mathbb{F}^{n'}$ be a generic arithmetic verification circuit with depth d and number of gates S . There exists an ILPZK proof that proves the satisfiability of \mathcal{C} with the following features:*

- *The prover runs in time $\mathcal{O}(S)$.*
- *The verifier runs in time $\mathcal{O}(n + n' + d^2 + d \log S + T)$, where $\mathcal{O}(T)$ is the time for evaluating MLEs.*
- *Round complexity is $\mathcal{O}(d \log S)$.*
- *Proof length is $\mathcal{O}(n + d \log S + d^2)$.*
- *Soundness error is $\mathcal{O}(\frac{d \log S}{|\mathbb{F}|})$.*

We remark that here the $\mathcal{O}(d^2)$ term in the proof length is always upper bounded by $\mathcal{O}(S)$. And only in some extremely bad cases (e.g., a narrow circuit with each layer connected to all its previous layers), the upper bound is reached.

Support Circuits Over Small Fields. Both of our ILPZK constructions for layered and generic circuits can be adapted to yield significant savings in proof

size when proving circuits over small fields (e.g. Boolean circuits) through the use of subfield VOLE. More concretely, taking the Boolean layered circuits for example, the proof size counted in bits is $\mathcal{O}(n + d \log S \log |\mathbb{F}|)$, shaving off a $\log |\mathbb{F}| = \mathcal{O}(\kappa)$ factor from the witness length n , where κ is the security parameter. This is a big advantage compared against the non-VOLE-based proof systems, as the usual way adapting such a protocol to work for small fields, is to view the small field computations as computations over its extension field, which not only incurs an overall $\mathcal{O}(\kappa)$ overhead, but also possibly demands attentions to guarantee that computations of circuits are indeed over the small field. On the other hand, our protocols still have significant asymptotic advantage compared to $\mathcal{O}(n + S)$ -bit communication of conventional VOLE-based ZK protocols (e.g., QuickSilver [36]).

NIZK from Compiling ILPZK with VOLE Protocols. Both of our ILPZK constructions for layered and generic circuits indeed satisfy additional properties of *public-coin* and *round-by-round soundness*, which allow us to squash interactions via the Fiat-Shamir transform. We first transform an ILPZK into an interactive VOLE-based ZK in the random VOLE-hybrid model, for which we prove security in UC-framework. Then we show that we can obtain designated verifier NIZK arguments from a pseudorandom correlation generator (PCG) instantiation of random VOLE. In addition, we can obtain publicly verifiable NIZK arguments from the VOLE-in-the-head (VOLEith) [3,27] technique. In general, the former has smaller computation, while the latter has smaller communication. This allows our constructions to find applications in a wider range of scenarios.

1.2 Technical Overview

We provide more details about how we achieve sublinear communication while maintaining linear prover computation in the context of VOLE-based ZK.

“Gate-by-gate” Limitations of LPZK. VOLE-based ZK proofs essentially lie in the scope of “commit-and-prove” paradigm, where VOLE serves as a commitment scheme. A vector \mathbf{x} is committed via VOLE in the sense that, the prover obtains random \mathbf{M} and the verifier obtains random \mathbf{K}, Δ such that $\mathbf{K} = \mathbf{x} \cdot \Delta + \mathbf{M}$, denoted by $[\mathbf{x}]$. VOLE naturally satisfies a linearly homomorphic property, which is the key to allowing to evaluate circuits underneath VOLE. For instance, given $[x], [y]$ for two values x, y and a scalar a , the commitment $[z]$ is obtained by \mathcal{P} setting $M_z := aM_x + M_y$ and \mathcal{V} setting $K_z := aK_x + K_y$, where $z := ax + y$.

Most conventional VOLE-based ZK protocols follow a “gate-by-gate” paradigm. We take LPZK [17] for example, as the works [16,17,32,36] differ slightly on low level details concerning how multiplication gates are verified. The prover first commits to the witness and all the intermediate values individually via VOLE as described above. The linear homomorphism property of VOLE implies that verification of addition gates can be realized for free, in the sense that the output commitment can be locally computed from input commitments. For multiplication gates, the prover needs to provide evidences supporting the claim that each multiplication gate is computed correctly, which is done by

appending additional entries to VOLE. It is not hard to see that witness, outputs of multiplication gates, and evidences for multiplications should all be included in the VOLE. This gives an intuition that the proof size has to be linear in the circuit size.

“Layer-by-layer” via Sum-check Protocol. Suppose for simplicity that we have a layered circuit. If we could commit to a whole layer of intermediate wire values using a small number of entries in a VOLE instance, and check relations layer-by-layer with a cost strictly smaller than verifying each gate, this should be sufficient for breaking the linear proof size barrier. This is in fact how the GKR [19] interactive proof protocol proceeds. We emphasise that no zero-knowledge is required there and the prover can reveal committed values directly to the verifier in plaintext. In a bare-bone sketch, GKR proceeds from output layer to input layer sequentially, and employs a sum-check protocol for the layer-by-layer reduction. In more detail, for each layer, GKR starts with a claim about the values of this layer, applies sum-check, and ends with a claim about the values of the previous layer. The final claim about the input layer is actually a claim about the witness, and in fact a much simpler statement to prove.

Cast in our ILPZK framework, and recall that the prover and the verifier are allowed to collectively generate an affine line $\mathbf{v}(x) := \mathbf{a} \cdot x + \mathbf{b}$, the whole affine line \mathbf{v} then can be intuitively divided into d blocks $\mathbf{v}^{(0)}(x), \mathbf{v}^{(1)}(x), \dots, \mathbf{v}^{(d-1)}(x)$, each specifying a “layer-by-layer” reduction. For simplicity, we call every $\mathbf{v}^{(i)}$ a sub-line of \mathbf{v} . Essentially, each sub-line consists of a part serves as commitments of sum-check messages, and a part that proves in zero-knowledge the commitments are honestly generated. These together convince \mathcal{V} that \mathcal{P} has generated a valid GKR proof that a GKR verifier would accept. As the length of each sub-line is sublinear in the number of gates in the corresponding layer, we indeed obtain an ILPZK with sublinear proof size as desired. The generic circuit case is similarly handled incorporating the recent breakthrough results from [37].

Linear Time Prover. The techniques for achieving a linear time prover in our ILPZK constructions for layered circuits and generic circuits follow closely from Libra [34] and Virgo++ [37], respectively. In addition to costs of running GKR in the clear, extra prover computational costs essentially come from proving all commitments of sum-check messages are correctly generated, which is linear (a small constant multiplicative overhead) in the number of sum-check messages. Therefore, our constructions maintain a linear time prover.

Support Arithmetic Circuits Over any Field. We next sketch how our ILPZK constructions can benefit from utilizing subfield VOLE. In a subfield VOLE instance $\mathbf{K} = \mathbf{x} \cdot \Delta + \mathbf{M}$, \mathbf{x} is over a small field \mathbb{F}_p , while $\mathbf{K}, \mathbf{M}, \Delta$ are over a large extension field \mathbb{F}_{p^r} . In fact, subfield VOLE can be viewed as a commitment scheme for elements of a subfield \mathbb{F}_p . Directly replacing the random VOLE in VOLE-based ZK protocols (e.g., QuickSilver [36]) with a random subfield VOLE allows to prove circuits over \mathbb{F}_p , with the same asymptotic communication but counted in the number of \mathbb{F}_p elements. However, for the affine line $\mathbf{v}(x) := \mathbf{a} \cdot x + \mathbf{b}$

in our ILPZK protocols, due to the use of MLEs in sum-check, part of \mathbf{a} entries are \mathbb{F}_{p^r} elements even though the arithmetic circuit is over \mathbb{F}_p . Intuitively, in order to apply the subfield VOLE techniques, we need an efficient construction of a mixture of standard VOLE and subfield VOLE, where they share the same random Δ . This seems a rather interesting variant of VOLE and might be useful in other application scenarios. We observe that to construct the desired variant of VOLE, it suffices to show that standard VOLE can be constructed from subfield VOLE. We provide a natural construction by fixing a basis of \mathbb{F}_{p^r} over \mathbb{F}_p .

1.3 Comparison with Related Works

On the one hand, our constructions are within the context of VOLE-based ZK. On the other hand, as we distill ideas from the IP literature, our constructions essentially follow the insightful idea of Cramer and Damgård [15], where they show how to obtain zero-knowledge arguments from IPs by using a cryptographic commitment scheme. We now briefly compare our techniques with related techniques from the literature and carefully argue the pros and cons.

Existing ZK Protocols with Linear Prover time and Sublinear Proof Size. As techniques used in IOP-based ZK protocols are quite different, we omit the detailed comparison. Compared to the state-of-the-art Brakedown [21], our constructions have lower computational overhead for small and medium-sized circuits, and have no restriction on the field size.

Essentially, our constructions share high-level similarities with ZK protocols based on sum-check, e.g., Hyrax [31], Libra [34], Virgo [38], Virgo++ [37], Spartan [28], and Cerberus [22]. In a high-level, the main difference is that here we use a lightweight linearly homomorphic commitment scheme (from VOLE) instead of a heavy one (Pedersen commitment [26]) in Hyrax, or a more powerful polynomial commitment based on FRI [6] in Virgo, or a pairing-based polynomial commitment in Libra and Virgo++, or a discrete-logarithm-based polynomial commitment in Spartan⁴, or a Ligerio-based polynomial commitment [1] in Cerberus⁵. This brings us the following features.

In addition to standalone-security, our constructions are statistically UC-secure in the random VOLE-hybrid model. As random VOLE correlations can be efficiently generated either from learning parity with noise (LPN) assumption [7] or assuming a pseudo-random generator (PRG), our ZK protocols are post-quantum when implemented. Finally, our constructions natively support proving statements over arbitrary-sized fields, by using the subfield VOLE technique. To our best knowledge, no previous protocol has achieved all these features simultaneously in the context of sum-check-based ZK.

For circuits over sufficiently large fields, our constructions have almost the same asymptotic performance as Libra and Virgo++. Our constructions instead use a light-weight VOLE-based commitment scheme. A consequence is that,

⁴ Spartan actually has prover computation $\mathcal{O}_\kappa(|\mathcal{C}|)$.

⁵ Cerberus can achieve linear prover time by using linear-time encodable codes, while they implement with Reed-Solomon codes, leading to a quasi-linear time prover.

when the circuit size is not significantly larger than the witness size, our constructions have slightly larger proof size, while they have larger computational overhead. For Boolean circuits, we offer much better concrete efficiency than Libra and Virgo++ in both communication and computation.

VOLE-based ZK. Wolverine [32] was the first VOLE-based ZK that works for arbitrary-sized fields with communication of 4 field elements per multiplication gate. The work LPZK [17] and its follow-up work QuickSilver [36] reduced the communication to 1 field element per multiplication. Next, improved LPZK [16] showed that the communication can be further reduced by half when considering layered circuits. All these works proceed an arithmetic circuit in a “gate-by-gate” flavor, from which they gain benefits of linear prover time and small memory. However, this also incurs proof size inherently linear in the circuit size. Compared to these works, our constructions instead have a “layer-by-layer” flavor, and in general have significantly smaller communication, at the cost of increasing prover computation up to roughly $2\times$, memory increased to $\mathcal{O}(|\mathcal{C}|)$, and round complexity increased to $\mathcal{O}(d \log |\mathcal{C}|)$.

Several works consider optimizing communication complexity in special cases. Mac’n’Cheese [5] focused on proving the disjunction of statements, with communication cost only proportional to the longest one. QuickSilver [36] also proposed a VOLE-based ZK for proving the computation of multiple polynomials, with communication cost linear to the highest degree of these polynomials. AntMan [33] started with a construction that allows for simultaneously proving B evaluations of a circuit \mathcal{C} , with communication of $\mathcal{O}(B + |\mathcal{C}|)$ field elements, and prover computation of $\mathcal{O}(B|\mathcal{C}| \log B)$. Then, they showed how to turn this construction into a VOLE-based ZK for a general circuit case, which has communication complexity $\mathcal{O}(|\mathcal{C}|^{3/4})$, and prover computation $\mathcal{O}(|\mathcal{C}| \log |\mathcal{C}|)$.

Compared to AntMan (for general circuits), our construction has linear prover time, smaller communication in most application scenarios, the same asymptotic memory consumption, and more rounds. Besides, our construction is information-theoretic in the random VOLE-hybrid model and is public-coin, while AntMan relies on an additively-homomorphic encryption scheme and is not public-coin.

2 Preliminaries

Notations. In this paper, bold letters (e.g. $\mathbf{a}, \mathbf{b}, \mathbf{M}, \mathbf{K}$) are used to denote vectors. Besides, we use x_i to denote the i_{th} -component of the vector \mathbf{x} . We use $[a, b]$ (or $[a, b + 1)$ sometimes) to denote the set of integers in the range from a to b . A commitment of some value x is denoted by $[x]$. We use $x \stackrel{\$}{\leftarrow} \mathbb{F}$ to denote that x is uniformly sampled from a field \mathbb{F} . We identify the set $\{0, 1\}^\ell$ and the set $[0, 2^\ell)$ through the natural bijection between the two sets.

Security Model, Functionalities, and Missing Proofs. We provide security proofs of our ZK protocols in the universal composability (UC) framework [13], in which we formally define a zero-knowledge functionality \mathcal{F}_{ZK} . In particular,

we consider active adversary and static corruption. Due to space constraints, we refer detailed security proofs and more preliminaries to the full version of this paper [23].

2.1 VOLE-Based Commitment

Random vector oblivious linear evaluation (VOLE) is a functionality that allows two parties P_S, P_R to obtain random correlated values. In more detail, the sender P_S obtains two vectors \mathbf{M}, \mathbf{x} , while the receiver P_R obtains a scalar Δ and a vector \mathbf{K} such that $\mathbf{K} = \mathbf{M} + \mathbf{x} \cdot \Delta$. We formalize the ideal functionality of random VOLE over finite field \mathbb{F} in Fig. 1.

The above VOLE correlation naturally induces a commitment scheme over \mathbb{F} . In the commit phase, the values \mathbf{x} are committed in the sense that through a VOLE functionality the sender obtains \mathbf{M} , and the receiver obtains Δ, \mathbf{K} , such that $\mathbf{K} = \mathbf{M} + \mathbf{x} \cdot \Delta$, denoted by $[\mathbf{x}]$. In the unveil phase, $[\mathbf{x}]$ are opened by the sender sending \mathbf{x}, \mathbf{M} to the receiver, who then checks $\mathbf{K} = \mathbf{M} + \mathbf{x} \cdot \Delta$. Such VOLE-based commitment schemes satisfy perfect hiding and statistical binding. Intuitively, the receiver learns nothing about \mathbf{x} before the unveil phase and the sender cannot open $[\mathbf{x}]$ to $\mathbf{x}' \neq \mathbf{x}$ unless he succeeds in guessing the Δ of the receiver.

It can be observed that the above commitment scheme also satisfies a linearly-homomorphic property. Given commitments $[x_1], \dots, [x_\ell]$ and public coefficients $c, c_1, \dots, c_\ell \in \mathbb{F}$, the two parties can locally compute $[y] = c + \sum_{i \in [\ell]} c_i \cdot [x_i]$ by setting $y = c + \sum_{i \in [\ell]} c_i \cdot x_i$, $M_y = \sum_{i \in [\ell]} c_i \cdot M_{x_i}$, and $K_y = \Delta \cdot c + \sum_{i \in [\ell]} c_i \cdot K_{x_i}$. In particular, we have $[y] = [x] + (y - x)$. This allows the sender to commit some y of his choice by sending $y - x$ to the receiver, given a commitment $[x]$ of a random x produced by a random VOLE functionality. From this observation, we also define a chosen-input VOLE functionality, which can be easily realized by a random VOLE functionality.

Functionality $\mathcal{F}_{\text{rVOLE}}^{\mathbb{F}}$

Init: Upon receiving (Init) from both parties, sample $\Delta \xleftarrow{\$} \mathbb{F}$ if P_R is honest, and receive $\Delta \in \mathbb{F}$ from the adversary \mathcal{A} otherwise. Store Δ and send it to P_R . All further (Init) commands will be ignored.

Extend: Upon receiving (Extend, N) from both parties, proceed as follows:

1. If P_R is honest, sample $\mathbf{K} \xleftarrow{\$} \mathbb{F}^N$. Otherwise receive \mathbf{K} from \mathcal{A} .
2. If P_S is honest, sample $\mathbf{x} \xleftarrow{\$} \mathbb{F}^N$ and compute $\mathbf{M} := \mathbf{K} - \Delta \cdot \mathbf{x} \in \mathbb{F}^N$. Otherwise, receive $\mathbf{x} \in \mathbb{F}^N$ and $\mathbf{M} \in \mathbb{F}^N$ from \mathcal{A} and then recompute $\mathbf{K} := \mathbf{M} + \Delta \cdot \mathbf{x}$.
3. Send (\mathbf{x}, \mathbf{M}) to P_S and \mathbf{K} to P_R .

Fig. 1. Ideal functionality for random VOLE over \mathbb{F} .

2.2 Multi-linear Extension

The multi-linear extension (MLE) plays a crucial role in the study of interactive proofs. We give a formal definition as follows:

Definition 1 (Multi-Linear Extension). *Let $f : \{0, 1\}^\ell \rightarrow \mathbb{F}$ be a function that maps the ℓ -dimensional binary hypercube to a field \mathbb{F} . The multi-linear extension of f is the unique polynomial $\tilde{f} : \mathbb{F}^\ell \rightarrow \mathbb{F}$ such that $\tilde{f}(x_1, \dots, x_\ell) = f(x_1, \dots, x_\ell)$ for all $x_1, \dots, x_\ell \in \{0, 1\}$, where the degree of \tilde{f} in each variable is 1. Moreover, \tilde{f} has the form*

$$\tilde{f}(x_1, \dots, x_\ell) = \sum_{\omega \in \{0,1\}^\ell} f(\omega) \cdot \chi_\omega(x_1, \dots, x_\ell),$$

where, for any $\omega = (\omega_1, \dots, \omega_\ell)$,

$$\chi_\omega(x_1, \dots, x_\ell) := \prod_{i=1}^\ell (x_i \omega_i + (1 - x_i)(1 - \omega_i)).$$

The set $\{\chi_\omega : \mathbb{F}^\ell \rightarrow \mathbb{F}\}_{\omega \in \{0,1\}^\ell}$ is referred to as the set of multi-linear Lagrange basis polynomials with interpolating set $\{0, 1\}^\ell$.

W.l.o.g., assume n is a power of two, then a vector $\mathbf{W} := (w_0, \dots, w_{n-1})$ over \mathbb{F} can be naturally viewed as a function $W : \{0, 1\}^{\log n} \rightarrow \mathbb{F}$ such that $W(i) = w_i$ for all $i \in [0, n)$. Hence, we define the multi-linear extension of a vector \mathbf{W} in this way, similarly denoted by \tilde{W} . To evaluate the multi-linear extension \tilde{W} of \mathbf{W} efficiently, we employ the algorithm proposed in [30], which takes $\mathcal{O}(n)$ time and $\mathcal{O}(n)$ memory usage.

Lemma 1 ([30]). *Assume $n = 2^\ell$ and given $\mathbf{W} \in \mathbb{F}^n$ and $\mathbf{r} \in \mathbb{F}^\ell$, one can compute $\tilde{W}(\mathbf{r})$ in $\mathcal{O}(n)$ time and $\mathcal{O}(n)$ space.*

2.3 Sum-Check Protocol and GKR Protocol

Our ILPZK proof systems distill ideas from the well-known GKR protocol [19], which involves a multi-variate sum-check protocol [25]. We overview the two protocols here.

Sum-check protocols are used to sum up polynomial evaluations on a specific set in a verifiable way, and play a crucial role in designing succinct arguments. We focus on multi-variate sum-check problems, which refer to, given some public ℓ -variate polynomial $f : \mathbb{F}^\ell \rightarrow \mathbb{F}$, the prover \mathcal{P} wants to convince the verifier \mathcal{V} such that

$$H = \sum_{b_1, \dots, b_\ell \in \{0,1\}} f(b_1, \dots, b_\ell),$$

without \mathcal{V} computing H by evaluating f at 2^ℓ points on her own. Assuming the maximum degree of f in each variable is d , the sum-check protocol has communication complexity $\mathcal{O}(d\ell)$, round complexity ℓ , and soundness error $\mathcal{O}(d\ell/|\mathbb{F}|)$. Moreover, if one uses a linear time algorithm of evaluating MLEs (e.g., Lemma 1), sum-check can be realized with linear prover time as shown in [29].

Lemma 2 ([29]). *Assume $n = 2^\ell$ and given ℓ -variate multi-linear polynomials $f_1, \dots, f_d : \mathbb{F}^\ell \rightarrow \mathbb{F}$. Applying sum-check on the ℓ -variate polynomial $g := f_1 \cdots f_d$ takes prover time $\mathcal{O}(dn)$.*

The GKR protocol is an interactive proof protocol, and can be used for evaluating layered arithmetic circuits in a verifiable way. More specifically, given a layered circuit \mathcal{C} and inputs \mathbf{w} known to both parties, through invoking GKR, \mathcal{P} can convince \mathcal{V} that h is indeed the evaluation of \mathcal{C} on \mathbf{w} without \mathcal{V} computing $\mathcal{C}(\mathbf{w})$ by herself. In a high level, GKR proceeds the circuit from output to input, in a layer-by-layer fashion. For each layer, GKR starts with a claim about values in this layer and employs a multi-variate sum-check protocol, which reduces the claim to claims about the previous layer. Then to proceed the previous layer, a *condensing* technique was designed in GKR that allows to combine multiple claims to one claim. The final claim about the input layer can be checked by \mathcal{V} directly. Assuming the layered circuit \mathcal{C} has depth d and S gates, GKR has communication complexity $\mathcal{O}(d \log S)$, round complexity $\mathcal{O}(d \log S)$, and soundness error $\mathcal{O}(d \log S / |\mathbb{F}|)$. In [34], GKR is optimized to have a linear prover time. Furthermore, if \mathcal{C} is log-space uniform, authors of [19] showed that the GKR verifier can run in time $\mathcal{O}(\log S)$.

2.4 LPZK [17] and QuickSilver [36]

The LPZK proof system essentially follows the “commit-and-prove” paradigm. Intuitively, the proof consists of two parts, where the first part serves as (linearly-homomorphic) commitments of the extended witness (i.e., wire values in the case of circuit satisfiability), and the second part proves (in zero-knowledge) that values underneath the commitments are exactly the extended witness. In the case of proving arithmetic circuit satisfiability, the extended witness consists of all wire values and it suffices to prove the satisfiability of a degree-2 relation dependent on the circuit topology. To start with, we show two simple examples of LPZK proof [17], which together allow to prove arbitrary degree-2 constraints.

1. **Linear constraint:** Let $\alpha, \beta \in \mathbb{F}$ be two coefficients known to each other, and suppose \mathcal{P} wants to convince \mathcal{V} that he holds some $a_1, a_2 \in \mathbb{F}$ such that $a_1 = \alpha \cdot a_2 + \beta$. The corresponding LPZK works as follows:
 - \mathcal{P} constructs an affine line $\mathbf{v}(x) := \mathbf{a} \cdot x + \mathbf{b}$ of dimension-3 with $v_1(x) := a_1 \cdot x + b_1$, $v_2(x) := a_2 \cdot x + b_2$ (as commitments of \mathbf{a}), and $v_3(x) := 0 \cdot x + b_3$, where $b_1, b_2 \stackrel{\$}{\leftarrow} \mathbb{F}$ and $b_3 = b_1 - \alpha \cdot b_2$.
 - Given an evaluation $\mathbf{v}(\Delta)$, \mathcal{V} checks that $v_3(\Delta) \stackrel{?}{=} v_1(\Delta) - \alpha \cdot v_2(\Delta) - \beta \cdot \Delta$ ⁶.
2. **Multiplicative constraint:** Suppose \mathcal{P} wants to convince \mathcal{V} that he holds some $a_1, a_2, a_3 \in \mathbb{F}$ such that $a_3 = a_1 \cdot a_2$. The corresponding LPZK is as follows:

⁶ We remark that a_3 must be 0, which can be guaranteed by \mathcal{P} sending b_3 to \mathcal{V} in the clear, which also shortens the VOLE length.

- \mathcal{P} constructs an affine line $\mathbf{v}(x) := \mathbf{a} \cdot x + \mathbf{b}$ of dimension-4 with $v_1(x) := a_1 \cdot x + b_1$, $v_2(x) := a_2 \cdot x + b_2$, $v_3(x) := a_3 \cdot x + b_3$ (as commitments of \mathbf{a}), and $v_4(x) := a_4 \cdot x + b_4$, where $b_1, b_2, b_3 \stackrel{\$}{\leftarrow} \mathbb{F}$, $a_4 := a_1 b_2 + a_2 b_1 - b_3$, and $b_4 := b_1 b_2$.
- Given an evaluation $\mathbf{v}(\Delta)$, \mathcal{V} checks that $v_1(\Delta) \cdot v_2(\Delta) \stackrel{?}{=} v_3(\Delta) \cdot \Delta + v_4(\Delta)$.

As any degree-2 constraint can be represented by combinations of linear constraints and multiplicative constraints, one can naturally extend the above two constructions and give a construction that proves arbitrary degree-2 relations. In addition, the completeness and security directly follow those of the underlying two simple constructions, on which we defer a detailed discussion to [23, Appendix A].

In this paper, we mainly focus on degree-2 relations consisting of individual t_1 linear constraints and t_2 multiplicative constraints with n inputs. For proving such degree-2 relations, the above LPZK construction requires the affine line $\mathbf{v}(x)$ to have length at least $n + t_1 + t_2$.

Batch-Check Linear/Multiplicative Constraints. Let \mathcal{C} be an arithmetic circuit over \mathbb{F} with n inputs, t multiplication gates, and arbitrary addition gates. For proving the satisfiability of \mathcal{C} , both LPZK [17] and QuickSilver [36] first transform it into a degree-2 relation consisting of one linear constraint and t multiplicative constraints with $n+t$ inputs. Thus, the length in LPZK is $n+2t+1$ ($n+t$ for “committing”, and $t+1$ for “proving”). QuickSilver is essentially a two-round ILPZK construction, which reduces the length to $n+t+1$ by introducing one additional round of interaction. Instead of building one sub-line per multiplicative constraint in LPZK, QuickSilver builds a sub-line that is the random linear combination of these t sub-lines. Similarly, the random linear combination technique also works for compressing linear constraints. Therefore, for proving a degree-2 relation consisting of individual t_1 linear constraints and t_2 multiplicative constraints with n inputs, it suffices to construct an affine line $\mathbf{v}(x)$ of length $n+2$ in the ILPZK setting.

3 Interactive Line-Point Zero-Knowledge Proof

In this section, we first give a formal definition of our new notion of Interactive LPZK proof system. Then we show how to compile such a proof system into a publicly verifiable NIZK argument via the VOLE-in-the-Head technique.

3.1 Defining ILPZK

We define interactive LPZK proof systems for arithmetic circuit satisfiability. The interactive LPZK proof system generalizes the original LPZK proof system, in the sense that an LPZK is essentially a 1-round interactive LPZK.

Definition 2 (ILPZK). *A t -round interactive line-point zero-knowledge (ILPZK) proof system for arithmetic circuit satisfiability over \mathbb{F} is given by a pair of algorithms (\mathbb{P}, \mathbb{V}) with the following syntax:*

- $\mathbb{P}(\mathcal{C}, \mathbf{w}, i, r_i)$ is a PPT algorithm that given an arithmetic verification circuit $\mathcal{C} : \mathbb{F}^n \rightarrow \mathbb{F}^{n'}$, a witness $\mathbf{w} \in \mathbb{F}^n$, a round index $i \in [1, t]$ and a sequence of strings r_i with each r_i being a prefix of r_{i+1} , outputs a pair of vectors $\mathbf{a}^{(i)}, \mathbf{b}^{(i)} \in \mathbb{F}^{\ell_i}$. Let $\ell := \sum_{i=1}^t \ell_i$ and $\mathbf{a} := (\mathbf{a}^{(1)}, \dots, \mathbf{a}^{(t)})$, $\mathbf{b} := (\mathbf{b}^{(1)}, \dots, \mathbf{b}^{(t)}) \in \mathbb{F}^\ell$, which specify an affine line $\mathbf{v}(x) = \mathbf{a} \cdot x + \mathbf{b}$ of dimension ℓ .
- $\mathbb{V}(\mathcal{C}, r_t, \Delta, \mathbf{v}_\Delta)$ is a polynomial time algorithm that, given a string r_t and an evaluation \mathbf{v}_Δ of the line $\mathbf{v}(x)$ at some point $\Delta \in \mathbb{F}$, outputs **accept** or **reject**.

With the above ILPZK algorithms, we formally define the ILPZK protocol.

ILPZK Protocol. Given a t -round ILPZK proof system with algorithms (\mathbb{P}, \mathbb{V}) as defined in Def. 2, and a chosen-input VOLE functionality $\mathcal{F}_{\text{VOLE}}$, there exists a t -round interactive protocol $\Pi(\mathbb{P}, \mathbb{V})$ proceeding as follows:

1. At the beginning, the verifier \mathcal{V} picks a $\Delta \in \mathbb{F}$, and sends it to $\mathcal{F}_{\text{VOLE}}^{\mathbb{F}}$. Let r_0 be an empty string.
2. In each round i , where $i = 1, 2, \dots, t$,
 - \mathcal{V} picks a string s_i and sends it to the prover \mathcal{P} . Then \mathcal{P} and \mathcal{V} compute r_i by appending s_i to the end of r_{i-1} .
 - \mathcal{P} runs $(\mathbf{a}^{(i)}, \mathbf{b}^{(i)}) \leftarrow \mathbb{P}(\mathcal{C}, \mathbf{w}, i, r_i)$. Then \mathcal{P} sends $(\mathbf{a}^{(i)}, \mathbf{b}^{(i)})$ to $\mathcal{F}_{\text{VOLE}}^{\mathbb{F}}$, which returns $\mathbf{v}_\Delta^{(i)} := \mathbf{a}^{(i)} \cdot \Delta + \mathbf{b}^{(i)}$ ⁷ to \mathcal{V} .
3. After t -round of interactions, \mathcal{V} already obtains $\mathbf{v}_\Delta := (\mathbf{v}_\Delta^{(1)}, \dots, \mathbf{v}_\Delta^{(t)})$. Finally, \mathcal{V} outputs $\mathbb{V}(\mathcal{C}, r_t, \Delta, \mathbf{v}_\Delta)$.

The ILPZK algorithms (\mathbb{P}, \mathbb{V}) should at least satisfy the following:

- **Perfect Completeness.** For any arithmetic circuit $\mathcal{C} : \mathbb{F}^n \rightarrow \mathbb{F}^{n'}$ and witness $\mathbf{w} \in \mathbb{F}^n$ such that $\mathcal{C}(\mathbf{w}) = \mathbf{1}$, the verifier \mathcal{V} in $\Pi(\mathbb{P}, \mathbb{V})$ outputs **accept** with probability 1, if the prover \mathcal{P} honestly follows $\Pi(\mathbb{P}, \mathbb{V})$.
- **ε -Soundness.** For every arithmetic circuit $\mathcal{C} : \mathbb{F}^n \rightarrow \mathbb{F}^{n'}$ such that $\mathcal{C}(\mathbf{w}) \neq \mathbf{1}$ for all $\mathbf{w} \in \mathbb{F}^n$, a malicious \mathcal{P} with arbitrary cheating strategies in $\Pi(\mathbb{P}, \mathbb{V})$ convinces \mathcal{V} with probability at most ε .
- **Perfect Zero-Knowledge.** There exists a PPT simulator Sim such that, for any arithmetic circuit $\mathcal{C} : \mathbb{F}^n \rightarrow \mathbb{F}^{n'}$, any witness $\mathbf{w} \in \mathbb{F}^n$ such that $\mathcal{C}(\mathbf{w}) = \mathbf{1}$, any r_t , and any $\Delta \in \mathbb{F}$, the output distribution of $\text{Sim}(\mathcal{C}, r_t, \Delta)$ is distributed identically to \mathbf{v}_Δ , where $\mathbf{v}(x)$ is the affine line generated in $\Pi(\mathbb{P}, \mathbb{V})$ with \mathcal{P} holding \mathbf{w} and \mathcal{V} sending r_t .

Furthermore, our ILPZK constructions also satisfy the following stronger notions:

Public-coin. An ILPZK proof with algorithms (\mathbb{P}, \mathbb{V}) is *public-coin*, if each bit of \mathcal{V} 's messages (i.e., r_t, Δ) in $\Pi(\mathbb{P}, \mathbb{V})$ is independently and uniformly random.

⁷ For our constructions in Sect. 4.1 and 5.1, we abuse the notation $\mathbf{v}^{(i)}$ for stage i , and denote by $\mathbf{v}^{(i,j)}$ the sub-line generated in round j of stage i .

ε -Knowledge Soundness. An ILPZK proof with algorithms (\mathbb{P}, \mathbb{V}) has *ε -knowledge soundness*, if there exists an efficient extractor Ext such that, for any arithmetic circuit $\mathcal{C} : \mathbb{F}^n \rightarrow \mathbb{F}^{n'}$, any r_t selected by \mathcal{V} , and any \mathbf{v} generated by (malicious) \mathcal{P} that makes \mathcal{V} accept with $> \varepsilon$ probability, $\text{Ext}(\mathcal{C}, r_t, \mathbf{v})$ outputs a valid witness \mathbf{w} .

3.2 Compiling ILPZK to NIZK

Given a t -round *public-coin* ILPZK proof system with algorithms (\mathbb{P}, \mathbb{V}) , we show how to compile it into a NIZK argument via a two-step approach. As already shown above, from algorithms (\mathbb{P}, \mathbb{V}) , we can immediately obtain an interactive protocol $\Pi(\mathbb{P}, \mathbb{V})$, assuming the existence of a chosen-input VOLE functionality. In sketch, the first step is to transform $\Pi(\mathbb{P}, \mathbb{V})$ into an interactive protocol $\Pi'(\mathbb{P}, \mathbb{V})$, with a sufficient number of random VOLE correlations generated at the very beginning (say, offline phase). In the second step, we instantiate random VOLE in two ways, yielding two different types of compiling.

In the first step, we rely on the linearly-homomorphism of VOLE. Recall that in each round i of $\Pi(\mathbb{P}, \mathbb{V})$, \mathcal{V} is supposed to learn $\mathbf{v}^{(i)}(\Delta)$. Assume \mathcal{P} holds random $\mathbf{x}^{(i)}, \mathbf{M}^{(i)}$, and \mathcal{V} holds random $\mathbf{K}^{(i)}$ such that $\mathbf{K}^{(i)} = \mathbf{x}^{(i)} \cdot \Delta + \mathbf{M}^{(i)}$. Let \mathcal{P} send $\delta^{(i)} := \mathbf{a}^{(i)} - \mathbf{x}^{(i)}$ and $\theta^{(i)} := \mathbf{b}^{(i)} - \mathbf{M}^{(i)}$ to \mathcal{V} , who then can compute

$$\begin{aligned} \mathbf{v}^{(i)}(\Delta) &:= \mathbf{K}^{(i)} + \delta^{(i)} \cdot \Delta + \theta^{(i)} \\ &= (\mathbf{x}^{(i)} \cdot \Delta + \mathbf{M}^{(i)}) + (\mathbf{a}^{(i)} - \mathbf{x}^{(i)}) \cdot \Delta + (\mathbf{b}^{(i)} - \mathbf{M}^{(i)}) \\ &= \mathbf{a}^{(i)} \cdot \Delta + \mathbf{b}^{(i)}, \end{aligned}$$

as desired. Since $\mathbf{x}^{(i)}, \mathbf{M}^{(i)}$ are uniformly random, \mathcal{V} learns nothing about $\mathbf{v}^{(i)}(x)$ except for $\mathbf{v}^{(i)}(\Delta)$. By applying the above transformation for every round of $\Pi(\mathbb{P}, \mathbb{V})$, we obtain a t -round public-coin ZK protocol $\Pi'(\mathbb{P}, \mathbb{V})$ in the random VOLE-hybrid model. We briefly state the security that $\Pi'(\mathbb{P}, \mathbb{V})$ could satisfy.

Malicious Security in UC-Framework. For our ILPZK constructions in this paper, not only the stand-alone security is satisfied, but also the UC-security. Formally speaking, for the ILPZK proof (\mathbb{P}, \mathbb{V}) in Sect. 4.1 or 5.1, the corresponding ZK protocol $\Pi'(\mathbb{P}, \mathbb{V})$ UC-realizes \mathcal{F}_{ZK} with malicious information-theoretic security in the random VOLE-hybrid model.

Below we introduce the two approaches of the second step.

Instantiating rVOLE with PCG. Boyle et al. [10, 12] introduced the cryptographic primitive of pseudorandom correlation generator (PCG), which is an extension of pseudorandom generator (PRG) from generating a batch of randomness to a batch of correlated randomness between some parties. PCG offers a concretely efficient candidate for generating random VOLE correlations in the offline phase. The authors of [11] presented a two-round maliciously secure construction of PCG for VOLE, and showed that one can obtain a designated verifier NIZK from combining it with a non-interactive online phase. Hence, by applying Fiat-Shamir transform to the online phase of $\Pi'(\mathbb{P}, \mathbb{V})$ and instantiating rVOLE with PCG, we can obtain a designated-verifier NIZK argument.

Applying VOLEitH. Given a public-coin rVOLE-based ZK protocol $\Pi'(\mathbb{F}, \mathbb{V})$ defined as above, one can typically obtain a publicly-verifiable NIZK argument through applying the VOLEitH technique proposed by Baum et al. [3]. In a high level, in the VOLEitH framework, random VOLE correlations are instantiated from a (two-round) all-but-one OT protocol, where the prover is the OT sender and the verifier the OT receiver. Then via the Fiat-Shamir transform, the interactive ZK protocol based on OT is turned into a non-interactive ZK⁸.

4 Interactive LPZK for Layered Arithmetic Circuits

In this section, we describe an ILPZK proof system for proving the satisfiability of layered arithmetic circuits, and we show that it achieves all properties as indicated in Theorem 1. In addition, based on this ILPZK, we are able to provide a self-contained VOLE-based ZK protocol $\Pi_{ZKl}^{\mathbb{F}}$ in Fig. 3. Finally, we prove that, when restricted to layered arithmetic circuits, our protocol $\Pi_{ZKl}^{\mathbb{F}}$ UC-realizes \mathcal{F}_{ZK} in the rVOLE-hybrid model with information-theoretic malicious security.

4.1 Our ILPZK Construction

Our ILPZK proof system for layered circuits is inspired by the well-known interactive proof protocol GKR [19], and we employ optimizations from Libra [34] to achieve a linear time prover. In a high level, the core idea of GKR is to reduce a claim about the output layer to a claim about the input layer in an iterative layer-by-layer sense. In [19], the authors discovered a brilliant equation that captures adjacent layers of the circuit, which allows the reduction to be done by employing a sum-check protocol on multi-variate polynomials [25]. For a better readability, let us first explicitly list some notations used in this section.

Notations. In this section, we consider as a layered (log-space uniform) arithmetic circuit over \mathbb{F} of depth d , size S , and fan-in two. Each layer of \mathcal{C} is labeled by a number from 0 to d , with 0 being the output layer and d being the input layer. More precisely, in a layered circuit \mathcal{C} , layer $i \in [0, d)$ consists of add/mult gates, which take input from outputs of layer $i + 1$, and layer d consists of input gates. W.l.o.g., we always assume each layer i of \mathcal{C} contains in total $s_i = 2^{k_i}$ gates (thus $S = \sum_{i=0}^{d-1} s_i$), and we label them in a pre-defined order. In addition, given the pre-defined order, we denote values on the output wires of gates in each layer i by $\mathbf{W}_i \in \mathbb{F}^{s_i}$. We also define two functions for each layer $i \in [0, d)$, $\text{add}_i, \text{mult}_i : \{0, 1\}^{k_i+2k_{i+1}} \rightarrow \{0, 1\}$, referred as “wiring predicates”. Each add_i (mult_i) takes one gate label $z \in \{0, 1\}^{k_i}$ in layer i and two gate labels $x, y \in \{0, 1\}^{k_{i+1}}$ in layer $i + 1$, and outputs 1 if and only if gate z in layer i is an addition (multiplication) gate that takes the output of gate x, y in layer $i + 1$ as

⁸ We remark that for statements defined over large fields, we need to apply the so-called *subspace VOLE* technique of [3] to significantly reduce the prover computation on generating random VOLE correlations. We omit the details as they are not the focus of this work, and refer to [3, 27].

input. We view each $\mathbf{W}_i \in \mathbb{F}^{s_i}$ as a function $W_i : \{0, 1\}^{k_i} \rightarrow \mathbb{F}$, and denote the multi-linear extension of W_i , mult_i , add_i by \widetilde{W}_i , $\widetilde{\text{mult}}_i$, $\widetilde{\text{add}}_i$, respectively.

With W_i , mult_i , add_i defined as above, it holds for all $i \in [0, d)$ that

$$W_i(z) = \sum_{x, y \in \{0, 1\}^{k_{i+1}}} \text{mult}_i(z, x, y)W_{i+1}(x)W_{i+1}(y) + \text{add}_i(z, x, y)(W_{i+1}(x) + W_{i+1}(y)),$$

where $z \in \{0, 1\}^{k_i}$. This immediately implies the following equation

$$\widetilde{W}_i(z) = \sum_{x, y \in \{0, 1\}^{k_{i+1}}} \widetilde{\text{mult}}_i(z, x, y)\widetilde{W}_{i+1}(x)\widetilde{W}_{i+1}(y) + \widetilde{\text{add}}_i(z, x, y)(\widetilde{W}_{i+1}(x) + \widetilde{W}_{i+1}(y)), \tag{1}$$

holds for all $z \in \mathbb{F}^{k_i}$.

Our ILPZK is also within the commit-and-prove paradigm. In the “commit” phase, the prover commits to the values that capture the full evaluation of $\mathcal{C}(\mathbf{w})$ (also depend on messages received from the verifier), while in the “prove” phase, the prover “opens” those commitments in zero-knowledge, thus the verifier can check whether the prover holds a witness \mathbf{w} such that $\mathcal{C}(\mathbf{w}) = \mathbf{1}$.

In a high level, our ILPZK proof for layered circuits can be divided into d sequential stages, each consisting of a “sub-commit” phase and a “sub-prove” phase (jumping ahead, each stage contains several rounds in our constructions). Intuitively, in each stage $i \in [0, d)$, the prover \mathcal{P} proves a sub-statement (indexed by i) to the verifier \mathcal{V} that he knows some \widetilde{W}_{i+1} such that Eq.(1) holds for \widetilde{W}_i evaluated at a random point $\mathbf{r}_i \in \mathbb{F}^{k_i}$ chosen by \mathcal{V} . However, the sub-statement i is never completely proved, unless \mathcal{P} proves the sub-statement $i + 1$. Until they reach stage $d - 1$, \mathcal{P} completely prove to \mathcal{V} that he knows the witness \widetilde{W}_d such that Eq.(1) holds for \widetilde{W}_{d-1} evaluated at a random point $\mathbf{r}_{d-1} \in \mathbb{F}^{k_{d-1}}$ chosen by \mathcal{V} . Eventually, these d sub-proofs together convince \mathcal{V} that \mathcal{P} knows a witness \mathbf{w} such that $\mathcal{C}(\mathbf{w}) = \mathbf{1}$.

Since the underlying commitment scheme is statistical binding, the sub-prove phase of each stage $i \in [0, d)$ can be deferred to stage $d - 1$ (i.e., when all sub-commit phases are completed). Hence, let us first describe how \mathcal{P} and \mathcal{V} proceed in the sub-commit phase of each stage i , and explain the arithmetic constraints that values underneath the commitments should satisfy. We remark that in fact, details of the first and the last stages are slightly different from the rest of the stages, which we will explain later. Essentially, \mathcal{P} and \mathcal{V} perform a two-phase (linear time) sum-check protocol on Eq.(1) underneath the commitment in each stage i .

In Each Stage $i \in [0, d)$, suppose that \mathcal{P} and \mathcal{V} start with an agreement on \mathbf{r}_i and a commitment $v_i := \widetilde{W}_i(\mathbf{r}_i) \cdot \Delta + b_i$ (with \mathcal{P} holds $\widetilde{W}_i(\mathbf{r}_i)$, b_i and \mathcal{V} holds v_i, Δ), denoted by $[\widetilde{W}_i(\mathbf{r}_i)]$, where $\mathbf{r}_i \in \mathbb{F}^{k_i}$ is a random point selected by \mathcal{V} (jumping ahead, \mathbf{r}_i is determined over several rounds of interactions during the previous stage). At the end of stage i , they will agree on a point $\mathbf{r}_{i+1} \in \mathbb{F}^{k_{i+1}}$ and obtain a commitment $v_{i+1} := \widetilde{W}_{i+1}(\mathbf{r}_{i+1}) \cdot \Delta + b_{i+1}$, denoted by $[\widetilde{W}_{i+1}(\mathbf{r}_{i+1})]$, so that they can move to the next stage.

According to Eq. (1), \mathcal{P} can define a $2k_{i+1}$ -variate polynomial

$$f_{r_i}^{(i)}(\mathbf{X}, \mathbf{Y}) := \widetilde{\text{mult}}_i(r_i, \mathbf{X}, \mathbf{Y}) \widetilde{W}_{i+1}(\mathbf{X}) \widetilde{W}_{i+1}(\mathbf{Y}) + \widetilde{\text{add}}_i(r_i, \mathbf{X}, \mathbf{Y}) (\widetilde{W}_{i+1}(\mathbf{X}) + \widetilde{W}_{i+1}(\mathbf{Y})). \quad (2)$$

Essentially, \mathcal{P} wants to convince \mathcal{V} that $\sum_{x,y \in \{0,1\}^{k_{i+1}}} f_{r_i}^{(i)}(x, y) = \widetilde{W}_i(r_i)$. We let \mathcal{P} and \mathcal{V} interact $2k_{i+1}$ rounds to capture the summation of $f_{r_i}^{(i)}(\mathbf{X}, \mathbf{Y})$ evaluated on the binary cube. Intuitively in each round, they sum-up one variable of $f_{r_i}^{(i)}(\mathbf{X}, \mathbf{Y})$. For the first k_{i+1} rounds, \mathcal{P} defines two univariate polynomials

$$A_{r_i}(\mathbf{X}) := \sum_{y \in \{0,1\}^{k_{i+1}}} \widetilde{\text{mult}}_i(r_i, \mathbf{X}, y) \cdot \widetilde{W}_{i+1}(y) + \widetilde{\text{add}}_i(r_i, \mathbf{X}, y),$$

and

$$B_{r_i}(\mathbf{X}) := \sum_{y \in \{0,1\}^{k_{i+1}}} \widetilde{\text{add}}_i(r_i, \mathbf{X}, y) \cdot \widetilde{W}_{i+1}(y).$$

This immediately implies that

$$\sum_{x,y \in \{0,1\}^{k_{i+1}}} f_{r_i}^{(i)}(x, y) = \sum_{x \in \{0,1\}^{k_{i+1}}} A_{r_i}(x) \cdot \widetilde{W}_{i+1}(x) + B_{r_i}(x).$$

In round j^9 , where $j = 1, \dots, k_{i+1}$, \mathcal{P} can compute a univariate polynomial $g^{(i,j)}(X_j)$ from $A_{r_i}(\mathbf{X})$, $\widetilde{W}_{i+1}(\mathbf{X})$, $B_{r_i}(\mathbf{X})$ as follows:

$$\begin{aligned} g^{(i,j)}(X_j) &:= \sum_{x_{j+1}, \dots, x_{k_{i+1}} \in \{0,1\}} (A_{r_i} \cdot \widetilde{W}_{i+1} + B_{r_i})(\bar{x}_1^{(i)}, \dots, \bar{x}_{j-1}^{(i)}, X_j, x_{j+1}, \dots, x_{k_{i+1}}) \\ &= \sum_{x_{j+1}, \dots, x_{k_{i+1}} \in \{0,1\}} \sum_{y \in \{0,1\}^{k_{i+1}}} f_{r_i}^{(i)}(\bar{x}_1^{(i)}, \dots, \bar{x}_{j-1}^{(i)}, X_j, x_{j+1}, \dots, x_{k_{i+1}}, y), \end{aligned} \quad (3)$$

where $\bar{x}_1^{(i)}, \dots, \bar{x}_{j-1}^{(i)} \in \mathbb{F}$ are sent by \mathcal{V} in the previous $j-1$ rounds. Since $A_{r_i}(\mathbf{X})$ and $\widetilde{W}_{i+1}(\mathbf{X})$ are multi-linear polynomials, $g^{(i,j)}(X_j)$ has degree 2. Then \mathcal{P} computes three coefficients of $g^{(i,j)}(X_j)$, denoted by $\mathbf{g}^{(i,j)} := (g_0^{(i,j)}, g_1^{(i,j)}, g_2^{(i,j)})$, and samples $\mathbf{b}^{(i,j)} \stackrel{\$}{\leftarrow} \mathbb{F}^3$. The interaction of round j proceeds as follows:

- \mathcal{P} sends $(\mathbf{g}^{(i,j)}, \mathbf{b}^{(i,j)})$ to $\mathcal{F}_{\text{VOLE}}^{\mathbb{F}}$, which then returns $\mathbf{v}_{\Delta}^{(i,j)} := \mathbf{g}^{(i,j)} \cdot \Delta + \mathbf{b}^{(i,j)}$ to \mathcal{V} .
- Upon receiving $\mathbf{v}_{\Delta}^{(i,j)}$, \mathcal{V} sends $\bar{x}_j^{(i)} \stackrel{\$}{\leftarrow} \mathbb{F}$ to \mathcal{P} .

Essentially, \mathcal{P} commits to a polynomial $g^{(i,j)}(X_j)$, denoted by $\llbracket g^{(i,j)}(\cdot) \rrbracket$, from which they can obtain $\llbracket g^{(i,j)}(\alpha) \rrbracket$ for any given $\alpha \in \mathbb{F}$. This is crucial for the sub-prove phase, and below we show arithmetic constraints on these $\llbracket g^{(i,j)}(\cdot) \rrbracket$. Recall that in the first round, \mathcal{P} builds an affine line $\mathbf{v}^{(i,1)}(x) := \mathbf{g}^{(i,1)} \cdot x + \mathbf{b}^{(i,1)}$, where $\mathbf{g}^{(i,1)}$ contains coefficients of $g^{(i,1)}(X_1)$. By definitions of $g^{(i,1)}, \widetilde{W}_i$, we

⁹ Here the interaction where \mathcal{P} first sends a line, and then \mathcal{V} replies a challenge is viewed as one round, which is consistent with our ILPZK definition (Def. 2).

have that $\widetilde{W}_i(\mathbf{r}_i) = g^{(i,1)}(0) + g^{(i,1)}(1)$, which gives a linear constraint: given $[\widetilde{W}_i(\mathbf{r}_i)], \llbracket g^{(i,1)}(\cdot) \rrbracket$, \mathcal{P} needs to convince \mathcal{V} that

$$\widetilde{W}_i(\mathbf{r}_i) = 2g_0^{(i,1)} + g_1^{(i,1)} + g_2^{(i,1)}.$$

In round $j \in [2, k_{i+1}]$, \mathcal{P} builds an affine line $\mathbf{v}^{(i,j)}(x) := \mathbf{g}^{(i,j)} \cdot x + \mathbf{b}^{(i,j)}$. By definitions of $g^{(i,j-1)}(X_{j-1}), g^{(i,j)}(X_j)$ (Eq.(3)), we have that $g^{(i,j-1)}(\bar{x}_{j-1}^{(i)}) = g^{(i,j)}(0) + g^{(i,j)}(1)$, which gives a linear constraint: given $\llbracket g^{(i,j-1)}(\cdot) \rrbracket, \llbracket g^{(i,j)}(\cdot) \rrbracket$, \mathcal{P} needs to convince \mathcal{V} that

$$g_0^{(i,j-1)} + g_1^{(i,j-1)} \cdot \bar{x}_{j-1}^{(i)} + g_2^{(i,j-1)} \cdot (\bar{x}_{j-1}^{(i)})^2 = 2g_0^{(i,j)} + g_1^{(i,j)} + g_2^{(i,j)}.$$

Now we move on to round $k_{i+1} + j$, where $j = 1, \dots, k_{i+1}$, \mathcal{P} instead computes a univariate polynomial $h^{(i,j)}(Y_j)$ with degree-2 simply from $f_{\mathbf{r}_i}^{(i)}(\mathbf{X}, \mathbf{Y})$:

$$h^{(i,j)}(Y_j) := \sum_{y_{j+1}, \dots, y_{k_{i+1}} \in \{0,1\}} f_{\mathbf{r}_i}^{(i)}(\bar{\mathbf{x}}^{(i)}, \bar{y}_1^{(i)}, \dots, \bar{y}_{j-1}^{(i)}, Y_j, y_{j+1}, \dots, y_{k_{i+1}}), \quad (4)$$

where $\bar{\mathbf{x}}^{(i)}, \bar{y}_1^{(i)}, \dots, \bar{y}_{j-1}^{(i)}$ are sent by \mathcal{V} in the previous $(k_{i+1} + j - 1)$ rounds. Similarly, \mathcal{P} computes $\mathbf{h}^{(i,j)}$ and samples $\mathbf{b}^{(i,k_{i+1}+j)} \stackrel{\$}{\leftarrow} \mathbb{F}^3$. The interaction of round $(k_{i+1} + j)$ proceeds as follows:

- \mathcal{P} sends $(\mathbf{h}^{(i,j)}, \mathbf{b}^{(i,k_{i+1}+j)})$ to $\mathcal{F}_{\text{VOLE}}^{\mathbb{F}}$, which then returns to \mathcal{V} a $\mathbf{v}_{\Delta}^{(i,k_{i+1}+j)} := \mathbf{h}^{(i,j)} \cdot \Delta + \mathbf{b}^{(i,k_{i+1}+j)}$.
- Upon receiving $\mathbf{v}_{\Delta}^{(i,k_{i+1}+j)}$, \mathcal{V} sends $\bar{y}_j^{(i)} \stackrel{\$}{\leftarrow} \mathbb{F}$ to \mathcal{P} .

Intuitively, \mathcal{P} commits to polynomials $h^{(i,j)}(Y_j)$ for $j \in [1, k_{i+1}]$, and below we show arithmetic constraints on these $\llbracket h^{(i,j)}(\cdot) \rrbracket$. Very similarly, in the round $(k_{i+1} + 1)$, we have the following linear constraint: given $\llbracket g^{(i,k_{i+1})}(\cdot) \rrbracket, \llbracket h^{(i,1)}(\cdot) \rrbracket$, \mathcal{P} needs to convince \mathcal{V} that

$$g_0^{(i,k_{i+1})} + g_1^{(i,k_{i+1})} \cdot \bar{x}_{k_{i+1}}^{(i)} + g_2^{(i,k_{i+1})} \cdot (\bar{x}_{k_{i+1}}^{(i)})^2 = 2h_0^{(i,1)} + h_1^{(i,1)} + h_2^{(i,1)}.$$

And in the following round $(k_{i+1} + j)$, where $j \in [2, k_{i+1}]$, we have the following linear constraint: given $\llbracket h^{(i,j-1)}(\cdot) \rrbracket, \llbracket h^{(i,j)}(\cdot) \rrbracket$, \mathcal{P} needs to convince \mathcal{V} that

$$h_0^{(i,j-1)} + h_1^{(i,j-1)} \cdot \bar{y}_{j-1}^{(i)} + h_2^{(i,j-1)} \cdot (\bar{y}_{j-1}^{(i)})^2 = 2h_0^{(i,j)} + h_1^{(i,j)} + h_2^{(i,j)}.$$

Note that after round $2k_{i+1}$, the prover \mathcal{P} has committed to a polynomial $h^{(i,k_{i+1})}(Y_{k_{i+1}})$, which leads to a commitment $[h^{(i,k_{i+1})}(\bar{y}_{k_{i+1}}^{(i)})]$. By definition, we have $h^{(i,k_{i+1})}(\bar{y}_{k_{i+1}}^{(i)}) = f_{\mathbf{r}_i}^{(i)}(\bar{\mathbf{x}}^{(i)}, \bar{\mathbf{y}}^{(i)})$. Suppose \mathcal{P} has committed to $\widetilde{W}_{i+1}(\bar{\mathbf{x}}^{(i)})$, $\widetilde{W}_{i+1}(\bar{\mathbf{y}}^{(i)})$. According to Eq.(2), we have the following degree-2 constraint: given $[h^{i,k_{i+1}}(\bar{y}_{k_{i+1}}^{(i)})], [\widetilde{W}_{i+1}(\bar{\mathbf{x}}^{(i)})], [\widetilde{W}_{i+1}(\bar{\mathbf{y}}^{(i)})]$, \mathcal{P} needs to convince \mathcal{V} that

$$h^{(i,k_{i+1})}(\bar{y}_{k_{i+1}}^{(i)}) = \widetilde{\text{mult}}_i(\mathbf{r}_i, \bar{\mathbf{x}}^{(i)}, \bar{\mathbf{y}}^{(i)}) \cdot \widetilde{W}_{i+1}(\bar{\mathbf{x}}^{(i)}) \cdot \widetilde{W}_{i+1}(\bar{\mathbf{y}}^{(i)}) + \widetilde{\text{add}}_i(\mathbf{r}_i, \bar{\mathbf{x}}^{(i)}, \bar{\mathbf{y}}^{(i)}) \cdot (\widetilde{W}_{i+1}(\bar{\mathbf{x}}^{(i)}) + \widetilde{W}_{i+1}(\bar{\mathbf{y}}^{(i)})).$$

In order to move to the next stage, we could let \mathcal{P} and \mathcal{V} perform stage $i+1$ twice on respective inputs $[\widetilde{W}_{i+1}(\bar{\mathbf{x}}^{(i)})]$, $[\widetilde{W}_{i+1}(\bar{\mathbf{y}}^{(i)})]$. However, this naive approach will incur an exponential blow-up in the circuit depth d .

To avoid this issue, we adapt the technique proposed in GKR [19], which allows for combining the above two sub-statements to one sub-statement $i+1$. The strategy requires one more round of interaction, and employs an observation that $\bar{\mathbf{x}}^{(i)}, \bar{\mathbf{y}}^{(i)} \in \mathbb{F}^{k_{i+1}}$ determine an affine line $L^{(i)}$ such that $L^{(i)}(0) = \bar{\mathbf{x}}^{(i)}$ and $L^{(i)}(1) = \bar{\mathbf{y}}^{(i)}$. Then \mathcal{P} can obtain a univariate polynomial $q^{(i)}(\cdot)$ with degree- k_{i+1} by restricting $\widetilde{W}_{i+1}(\cdot)$ to $L^{(i)}$, which satisfies that $q^{(i)}(0) = \widetilde{W}_{i+1}(\bar{\mathbf{x}}^{(i)})$ and $q^{(i)}(1) = \widetilde{W}_{i+1}(\bar{\mathbf{y}}^{(i)})$. Therefore, by \mathcal{P} committing to $q^{(i)}(X)$, the two commitments $[\widetilde{W}_{i+1}(\bar{\mathbf{x}}^{(i)})]$, $[\widetilde{W}_{i+1}(\bar{\mathbf{y}}^{(i)})]$ can be replaced by $[q^{(i)}(0)]$, $[q^{(i)}(1)]$. We let \mathcal{P} compute $\mathbf{q}^{(i)}$ and sample $\mathbf{b}^{(i,2k_{i+1}+1)} \xleftarrow{\$} \mathbb{F}^{k_{i+1}+1}$. Now the additional round $2k_{i+1}+1$ proceeds as follows:

- \mathcal{P} sends $(\mathbf{q}^{(i)}, \mathbf{b}^{(i,2k_{i+1}+1)})$ to $\mathcal{F}_{\text{VOLE}}^{\mathbb{F}}$, which then returns to \mathcal{V} a $\mathbf{v}_{\Delta}^{(i,2k_{i+1}+1)} := \mathbf{q}^{(i)} \cdot \Delta + \mathbf{b}^{(i,2k_{i+1}+1)}$.
- Upon receiving $\mathbf{v}_{\Delta}^{(i,2k_{i+1}+1)}$, \mathcal{V} sends $r^{(i)} \xleftarrow{\$} \mathbb{F}$ to \mathcal{P} .

Now the corresponding degree-2 constraint is as follows: given $[[h^{(i,k_{i+1})}(\cdot)]]$, $[[q^{(i)}(\cdot)]]$, \mathcal{P} needs to convince \mathcal{V} that

$$\sum_{l=0}^2 h_l^{(i,k_{i+1})}(\bar{\mathbf{y}}_{k_{i+1}}^{(i)})^l = \widetilde{\text{mult}}_i(\mathbf{r}_i, \bar{\mathbf{x}}^{(i)}, \bar{\mathbf{y}}^{(i)})q_0^{(i)} \left(\sum_{l=0}^{k_{i+1}} q_l^{(i)} \right) + \widetilde{\text{add}}_i(\mathbf{r}_i, \bar{\mathbf{x}}^{(i)}, \bar{\mathbf{y}}^{(i)}) \left(q_0^{(i)} + \left(\sum_{l=0}^{k_{i+1}} q_l^{(i)} \right) \right)$$

As for moving to the next stage, they can set $\mathbf{r}_{i+1} := L^{(i)}(r^{(i)})$, with \mathcal{P}, \mathcal{V} locally computing

$$b_{i+1} := \sum_{j=0}^{k_{i+1}} b_j^{(i,2k_{i+1}+1)} \cdot (r^{(i)})^j, v_{i+1} := \sum_{j=0}^{k_{i+1}} v_{\Delta,j}^{(i,2k_{i+1}+1)} \cdot (r^{(i)})^j,$$

respectively. Note that it is supposed to hold that $v_{i+1} = \widetilde{W}_{i+1}(\mathbf{r}_{i+1}) \cdot \Delta + b_{i+1}$, (i.e., they can obtain $[\widetilde{W}_{i+1}(\mathbf{r}_{i+1})]$ in this way). For simplicity, we denote $(\mathbf{v}^{(i,1)}(x), \dots, \mathbf{v}^{(i,2k_{i+1}+1)}(x))$ by $\mathbf{v}^{(i)}(x)$.

In the first stage, \mathcal{P} and \mathcal{V} need to run a setup at first. According to the definition of ILPZK, \mathcal{V} picks a random $\Delta \in \mathbb{F}$ and sends it to $\mathcal{F}_{\text{VOLE}}^{\mathbb{F}}$. Jumping ahead, \mathcal{P} will be required to generate a line that captures the witness \mathbf{w} before he received challenges in the last stage. As the line is only dependent on the witness, we let \mathcal{P} do this at the very beginning of stage 0. Thus, the one round of interaction for setup proceeds as follows:

- \mathcal{P} samples $\mathbf{b}^{(0,0)} \xleftarrow{\$} \mathbb{F}^{s_a}$, and sends $(\mathbf{w}, \mathbf{b}^{(0,0)})$ to $\mathcal{F}_{\text{VOLE}}^{\mathbb{F}}$. Then $\mathcal{F}_{\text{VOLE}}^{\mathbb{F}}$ returns $\mathbf{v}_{\Delta}^{(0,0)} := \mathbf{w} \cdot \Delta + \mathbf{b}^{(0,0)}$ to \mathcal{V} .
- Upon receiving $\mathbf{v}_{\Delta}^{(0,0)}$, \mathcal{V} sends $\mathbf{r}_0 \xleftarrow{\$} \mathbb{F}^{k_0}$ to \mathcal{P} .

Note that as in this paper we focus on the circuit satisfiability problem, the output values of layer 0 should be all 1's (i.e., $\mathbf{W}_0 = \mathbf{1} \in \mathbb{F}^{s_0}$) as long as \mathcal{P} inputs a witness \mathbf{w} . Hence, we let \mathcal{V} locally set $v_0 = \widetilde{W}_0(\mathbf{r}_0) \cdot \Delta$, and \mathcal{P} locally set $b_0 = 0$. This completes the setup, and they can move on. For simplicity, we let $\mathbf{v}^{(0)}(x)$ also include $\mathbf{v}^{(0,0)}(x)$.

In the last stage, to complete the whole proof, \mathcal{P} remains to convince \mathcal{V} that all previous affine lines $\mathbf{v}^{(0)}(x), \dots, \mathbf{v}^{(d-1)}(x)$ are honestly generated (i.e., run the deferred sub-prove phases.). In addition to the arithmetic constraints we have specified above, there is an arithmetic constraint on $[\widetilde{\mathbf{W}}_d(\mathbf{r}_d)]$ (obtained in stage $d - 1$) and $[\mathbf{W}_d]$ (obtained in stage 0), where \mathcal{P} needs to convince \mathcal{V} that

$$\widetilde{W}_d(\mathbf{r}_d) = \sum_{\omega \in \{0,1\}^{k_d}} W_d(\omega) \cdot \chi_\omega(\mathbf{r}_d),$$

which is induced by the Lagrange interpolation of Def 1.

So far we have shown the arithmetic constraints that $\mathbf{v}^{(0)}(x), \dots, \mathbf{v}^{(d-1)}(x)$ need to satisfy are actually degree-2 constraints. Therefore, we can introduce one more round of interaction to efficiently check these constraints in a batch, as indicated in the end of Sect. 2.4. For simplicity, we let $\mathbf{v}^{(d-1)}(x)$ include the extra two entries for batch checking linear and multiplication constraints.

4.2 Complexity

Here we analyze the complexity of our construction in Sect. 4.1.

Round Complexity. Our construction can be divided into d stages, with each stage $i \in [1, d - 2]$ having $2k_{i+1} + 1$ rounds. In particular, stage 0 has $2k_1 + 2$ rounds, and stage $d - 1$ has $2k_d + 2$ rounds. Thus, there are $2 + \sum_{i=1}^d (k_i + 1) = \mathcal{O}(d \log S)$ rounds in total.

Proof Size. The proof size is the summation of length of $\mathbf{v}^{(0)}(x), \dots, \mathbf{v}^{(d-1)}(x)$. For $i \in [1, d - 2]$, each $\mathbf{v}^{(i)}$ has length $6k_{i+1} + k_{i+1} + 1 = 7k_{i+1} + 1$, while $\mathbf{v}^{(0)}(x)$ has length $s_d + 7k_1 + 1$, and $\mathbf{v}^{(d-1)}(x)$ has length $7k_d + 1 + 2$. Thus, the total proof size is $\mathcal{O}(s_d + 2 + \sum_{i=1}^d (7k_i + 1)) = \mathcal{O}(n + d \log S)$.

Prover time. Applying “sum-check” dominates the prover time. By Lemma 2, for each stage i , performing a two-phase sum-check for generating sub-statements costs prover time $2 \sum_{j=0}^{k_{i,i+1}} \mathcal{O}(2^{k_{i,i+1}-j}) = \mathcal{O}(2^{k_{i,i+1}}) = \mathcal{O}(s_{i,i+1})$, and performing a sum-check for combining sub-statements costs prover time $\sum_{j=0}^{k_{i+1}} \mathcal{O}(2^{k_{i+1}-j}) = \mathcal{O}(2^{k_{i+1}}) = \mathcal{O}(s_{i+1})$. Thus, it takes the prover overall $\mathcal{O}(S)$ time.

Verifier Time. The verifier \mathcal{V} needs to at least read the entire proof, which takes time $\mathcal{O}(n + d \log S)$. In addition, \mathcal{V} also needs to evaluate multi-linear polynomials at some specific points, including computing $[\widetilde{W}_0(\mathbf{r}_0)]$, $[\widetilde{W}_d(\mathbf{r}_d)]$ and $\widetilde{\text{mult}}_i(\mathbf{r}_i, \widetilde{\mathbf{x}}^{(i)}, \widetilde{\mathbf{y}}^{(i)})$, $\widetilde{\text{add}}_i(\mathbf{r}_i, \widetilde{\mathbf{x}}^{(i)}, \widetilde{\mathbf{y}}^{(i)})$ for each $i \in [0, d]$. By Lemma 1, the

former two computations cost time in total $\mathcal{O}(n + s_0)$. Evaluating $\widetilde{\text{mult}}_i$ and $\widetilde{\text{add}}_i$ can be done in time $\mathcal{O}(\log S)$ for several types of circuits [14, 29], as they are usually very sparse. Also, [19] proposed a method for log-space uniform circuit, which takes verifier time $\mathcal{O}(d \log S)$ by outsourcing the computation to the prover. In summary, for layered log-space uniform circuits, \mathcal{V} runs in time $\mathcal{O}(n + s_0 + d \log S)$.

4.3 Security Proof in UC-Framework

We present in Fig. 3 a self-contained zero-knowledge protocol $\Pi_{\text{ZKI}}^{\mathbb{F}}$ in the random VOLE-hybrid model, which is based on our ILPZK proof in Sect. 4.1.

In the offline phase, the prover \mathcal{P} and the verifier \mathcal{V} invoke the random VOLE functionality $\mathcal{F}_{\text{rVOLE}}^{\mathbb{F}}$, and they obtain a certain number of random VOLE correlations (i.e., VOLE commitments of random values). Due to the linearity of VOLE correlations, \mathcal{P} can commit to a value w by sending $\delta := w - \nu$ to \mathcal{V} and computing $[w] := [\nu] + \delta$, where $[\nu]$ is a random VOLE correlation.

In the online phase, \mathcal{P} and \mathcal{V} follow the instructions of our ILPZK construction except that \mathcal{P} commits to values through random VOLE instead of VOLE. For a cleaner presentation, we design two tailored procedures for batch-checking linear and multiplicative constraints, with details in Fig. 2. In the case of checking linear constraints, it suffices to allow \mathcal{V} to check the value underneath a certain commitment is zero. More specifically, given a VOLE-based commitment $[x]$ with $K_x = x \cdot \Delta + M_x$, \mathcal{V} can check $[0] \stackrel{?}{=} [x]$ by \mathcal{P} revealing M_x and checking $K_x \stackrel{?}{=} M_x$. Hence, applying this procedure would not increase the number of random VOLE correlations. While for the case of checking multiplicative constraints, they need to consume one random VOLE correlation. More specifically, recall the multiplicative constraint check in Sect. 2.4, \mathcal{V} needs to obtain $v_4(\Delta)$ so that she can complete the verification. But directly revealing the two coefficients of $v_4(x)$ certainly leaks information about \mathcal{P} 's inputs. To prevent this leakage, one can mask it by an additional entry of random VOLE. Therefore, by using the random linear combination technique, performing the two checking procedures only consumes one additional entry of random VOLE.

The following theorem asserts the security of our protocol $\Pi_{\text{ZKI}}^{\mathbb{F}}$. Moreover, Theorem 3 implies Theorem 1.

Theorem 3. *Our ZK protocol $\Pi_{\text{ZKI}}^{\mathbb{F}}$ UC-realizes the ZK functionality \mathcal{F}_{ZK} in the $\mathcal{F}_{\text{rVOLE}}^{\mathbb{F}}$ -hybrid model with information-theoretic malicious security. In particular, the environment \mathcal{Z} 's advantage is $\mathcal{O}(\frac{d \log S}{|\mathbb{F}|})$.*

Extending to any Field. Recall that random subfield VOLE allows to commit elements of a small field \mathbb{F}_p over a large enough extension field \mathbb{F}_{p^r} . Therefore, to prove the satisfiability of a circuit \mathcal{C} over \mathbb{F}_p , one can substitute VOLE with subfield VOLE for “gate-by-gate” VOLE-based ZK protocols as all wire values are over \mathbb{F}_p . However, for our “layer-by-layer” protocol $\Pi_{\text{ZKI}}^{\mathbb{F}}$, the multi-linear extensions of layers must be evaluated at \mathbb{F}_{p^r} points for security guarantee (hence

Procedure Batch- Lin : on input $\{\llbracket m_i \rrbracket, \llbracket g^{(i,j)} \rrbracket, \llbracket h^{(i,j)} \rrbracket\}_{i \in [0,d], j \in [1, k_{i+1}]}$.

1. For $i \in [0, d-1]$, $j \in [1, k_{i+1}]$, \mathcal{P} and \mathcal{V} compute $([g^{(i,j)}(0)], [g^{(i,j)}(1)], [g^{(i,j)}(\bar{x}_j^{(i)})])$ and $([h^{(i,j)}(0)], [h^{(i,j)}(1)], [h^{(i,j)}(\bar{y}_j^{(i)})])$.
2. The verifier \mathcal{V} wants checks that $[0] \stackrel{?}{=} [m_i] - [g^{(i,1)}(0)] - [g^{(i,1)}(1)]$, $[0] \stackrel{?}{=} [g^{(i, k_{i+1})}(\bar{x}_{k_{i+1}}^{(i)})] - [h^{(i,1)}(0)] - [h^{(i,1)}(1)]$ and $[0] \stackrel{?}{=} [g^{(i,j)}(\bar{x}_j^{(i)})] - [g^{(i,j+1)}(0)] - [g^{(i,j+1)}(1)]$, $[0] \stackrel{?}{=} [h^{(i,j)}(\bar{y}_j^{(i)})] - [h^{(i,j+1)}(0)] - [h^{(i,j+1)}(1)]$ for $j \in [1, k_{i+1}]$. For simplicity, we naturally view these $N := 2 \sum_{i=1}^d k_i$ constraints as $([x_i], [y_i], [z_i])$ such that $z_i = x_i + y_i$ should hold for all $i \in [N]$.
3. \mathcal{V} samples $\lambda \stackrel{\$}{\leftarrow} \mathbb{F}^N$, and sends it to \mathcal{P} .
4. \mathcal{P} computes $M_{\text{lin}} := \sum_{i=1}^N \lambda_i (M_{z_i} - M_{x_i} - M_{y_i})$, and sends it to \mathcal{V} .
5. \mathcal{V} computes $K_{\text{lin}} := \sum_{i=1}^N \lambda_i (K_{z_i} - K_{x_i} - K_{y_i})$, and checks that $K_{\text{lin}} \stackrel{?}{=} M_{\text{lin}}$.

Procedure Batch- Mult : on input $\{\llbracket q^{(i)}(\cdot) \rrbracket, \llbracket h^{(i, k_{i+1})}(\cdot) \rrbracket\}_{i \in [0,d]}$.

1. For $i \in [0, d-1]$, \mathcal{P} and \mathcal{V} compute $[x_i] := [q^{(i)}(0)]$, $[y_i] := [q^{(i)}(1)]$, $[z_i] := [h^{(i, k_{i+1})}(\bar{y}_{k_{i+1}}^{(i)})]$, and $a_i := \widehat{\text{mult}}_i(\mathbf{r}_i, \bar{\mathbf{x}}^{(i)}, \bar{\mathbf{y}}^{(i)})$, $b_i := \widehat{\text{add}}_i(\mathbf{r}_i, \bar{\mathbf{x}}^{(i)}, \bar{\mathbf{y}}^{(i)})$.
2. \mathcal{V} wants to check that $z_i \stackrel{?}{=} a_i x_i y_i + b_i (x_i + y_i)$ for all i .
3. \mathcal{V} samples $\alpha \stackrel{\$}{\leftarrow} \mathbb{F}^d$ and sends it to \mathcal{P} .
4. They consume a random VOLE correlation $[\pi]$ in the sense that \mathcal{P} computes $M_{\text{mult}} := M_\pi + \sum_{i=0}^{d-1} \alpha_i (a_i M_{x_i} M_{y_i})$, and

$$x_{\text{mult}} := \pi + \sum_{i=0}^{d-1} \alpha_i (a_i (y_i M_{x_i} + x_i M_{y_i}) + b_i (M_{x_i} + M_{y_i}) - M_{z_i}),$$

while \mathcal{V} computes

$$K_{\text{mult}} := K_\pi + \sum_{i=0}^{d-1} \alpha_i (a_i K_{x_i} K_{y_i} + (b_i (K_{x_i} + K_{y_i}) - K_{z_i}) \cdot \Delta).$$

5. \mathcal{P} sends $(x_{\text{mult}}, M_{\text{mult}})$ to \mathcal{V} , who then checks that $K_{\text{mult}} \stackrel{?}{=} M_{\text{mult}} + x_{\text{mult}} \cdot \Delta$.

Fig. 2. Procedures for batch-checking linear and multiplicative constraints.

are over \mathbb{F}_{p^r} , while the witness \mathbf{w} is over \mathbb{F}_p . Therefore, we intuitively hope for a mixture of random VOLE and random subfield VOLE, where the verifier holds the same random Δ . We formalize the required functionality as $\mathcal{F}_{\text{sVOLE}}^{\mathbb{F}_p, \mathbb{F}_{p^r}}$. On top of $\mathcal{F}_{\text{sVOLE}}^{\mathbb{F}_p, \mathbb{F}_{p^r}}$, we can easily extend our ZK protocols to any field. It remains to show an efficient construction of $\mathcal{F}_{\text{sVOLE}}^{\mathbb{F}_p, \mathbb{F}_{p^r}}$. In fact, we observe that by fixing a basis of \mathbb{F}_{p^r} over \mathbb{F}_p , denoted by $\lambda_1, \dots, \lambda_r$, standard VOLE correlations can be locally computed from subfield VOLE correlations. For instance, given $[\mathbf{x}_1], \dots, [\mathbf{x}_r]$, where $\mathbf{x}_1, \dots, \mathbf{x}_r$ are over \mathbb{F}_p , the commitment of $\mathbf{x} := \sum_{i=1}^r \mathbf{x}_i \lambda_r$ over \mathbb{F}_{p^r} can

Protocol $\Pi_{\text{ZKl}}^{\mathbb{F}}$

Notations follow Sect. 4.1. The prover \mathcal{P} wants to convince the verifier \mathcal{V} that he holds a witness $\mathbf{w} \in \mathbb{F}^n$ such that $\mathcal{C}(\mathbf{w}) = 1$. **Offline phase**

1. The prover \mathcal{P} and the verifier \mathcal{V} send **(Init)** to $\mathcal{F}_{\text{rVOLE}}^{\mathbb{F}}$, and \mathcal{V} receives $\Delta \in \mathbb{F}$.
2. \mathcal{P} and \mathcal{V} send **(Extend, $n + \sum_{i=1}^d (7k_i + 1) + 1$)** to $\mathcal{F}_{\text{rVOLE}}^{\mathbb{F}}$, which returns commitments on random values, denoted by $[\nu]$, $[\mu]$, $[\pi]$, where $\nu \in \mathbb{F}^n$, $\pi \in \mathbb{F}$. For simplicity, we view μ as $\{\mu_{i,j}\}_{i \in [0,d], j \in [7k_{i+1}+1]}$.

Online phase

1. The prover \mathcal{P} and the verifier \mathcal{V} obtain $[\mathbf{w}]$ (by \mathcal{P} sending $\delta := \mathbf{w} - \nu$ to \mathcal{V}).
2. For each layer i , \mathcal{P} computes \mathbf{W}_i and stores them. \mathcal{V} sends a random $\mathbf{r}_0 \in \mathbb{F}^{k_0}$ to \mathcal{P} , then they locally compute $m_0 := \widetilde{W}_0(\mathbf{r}_0)$ (Note that $\mathbf{W}_0 = \mathbf{1} \in \mathbb{F}^{s_0}$).
3. For $i = 0, 1, \dots, d-1$,
 - (a) The prover \mathcal{P} defines the $2k_{i+1}$ -variate polynomial $f_{r_i}^{(i)}(\mathbf{X}, \mathbf{Y}) := \widetilde{\text{mult}}_i(\mathbf{r}_i, \mathbf{X}, \mathbf{Y}) \widetilde{W}_{i+1}(\mathbf{X}) \widetilde{W}_{i+1}(\mathbf{Y}) + \widetilde{\text{add}}_i(\mathbf{r}_i, \mathbf{X}, \mathbf{Y}) (\widetilde{W}_{i+1}(\mathbf{X}) + \widetilde{W}_{i+1}(\mathbf{Y}))$.
 - (b) For $j = 1, \dots, k_{i+1}$,
 - i. \mathcal{P} computes a univariate polynomial $g^{(i,j)}(X_j)$ of degree-2, writing as $g_0^{(i,j)} + g_1^{(i,j)} \cdot X_j + g_2^{(i,j)} \cdot X_j^2$. \mathcal{P} sends $g_0^{(i,j)} - \mu_{i,3j-2}$, $g_1^{(i,j)} - \mu_{i,3j-1}$, $g_2^{(i,j)} - \mu_{i,3j}$ to \mathcal{V} . They essentially obtain a triple of commitments $([g_0^{(i,j)}], [g_1^{(i,j)}], [g_2^{(i,j)}])$, denoted by $\llbracket g^{(i,j)}(\cdot) \rrbracket$.
 - ii. \mathcal{V} samples $\bar{x}_j^{(i)} \xleftarrow{\$} \mathbb{F}$ and sends it to \mathcal{P} .
 - (c) For $j = 1, \dots, k_{i+1}$,
 - i. \mathcal{P} computes a single variable polynomial $h^{(i,j)}(Y_j)$ of degree 2, writing as $h_0^{(i,j)} + h_1^{(i,j)} \cdot Y_j + h_2^{(i,j)} \cdot Y_j^2$. \mathcal{P} sends $h_0^{(i,j)} - \mu_{i,3k_{i+1}+3j-2}$, $h_1^{(i,j)} - \mu_{i,3k_{i+1}+3j-1}$, $h_2^{(i,j)} - \mu_{i,3k_{i+1}+3j}$ to \mathcal{V} . They essentially obtain a triple of commitments $([h_0^{(i,j)}], [h_1^{(i,j)}], [h_2^{(i,j)}])$, denoted by $\llbracket h^{(i,j)}(\cdot) \rrbracket$.
 - ii. \mathcal{V} samples $\bar{y}_j^{(i)} \xleftarrow{\$} \mathbb{F}$ and sends it to \mathcal{P} .
 - (d) Let $L^{(i)}$ be the unique line satisfying $L^{(i)}(0) = \bar{x}^{(i)}$, $L^{(i)}(1) = \bar{y}^{(i)}$. \mathcal{P} computes a univariate polynomial $q^{(i)}(X)$ by restricting \widetilde{W}_{i+1} to $L^{(i)}$, writing as $\sum_{j=0}^{k_{i+1}} q_j^{(i)} \cdot X^j$. \mathcal{P} sends $(q_0^{(i)} - \mu_{i,6k_{i+1}+1}, \dots, q_{k_{i+1}}^{(i)} - \mu_{i,7k_{i+1}+1})$ to \mathcal{V} , and similarly, they obtain $([q_0^{(i)}], \dots, [q_{k_{i+1}}^{(i)}])$, denoted by $\llbracket q^{(i)}(\cdot) \rrbracket$.
 - (e) \mathcal{V} selects $r^{(i)} \xleftarrow{\$} \mathbb{F}$ and sends it to \mathcal{P} . \mathcal{P} computes $m_{i+1} := q^{(i)}(r^{(i)})$. Then they set $\mathbf{r}_{i+1} := L^{(i)}(r^{(i)}) \in \mathbb{F}^{k_{i+1}}$, and compute $[m_{i+1}] := [q^{(i)}(r^{(i)})]$.
4. \mathcal{P} and \mathcal{V} perform the following checks.
 - (a) \mathcal{P} and \mathcal{V} run the procedure **Batch-Lin** in Fig. 2 on input tuples $\{([m_i], \llbracket g^{(i,j)}(\cdot) \rrbracket, \llbracket h^{(i,j)}(\cdot) \rrbracket)\}_{i \in [0,d], j \in [1, k_{i+1}]}$.
 - (b) \mathcal{P} and \mathcal{V} run the procedure **Batch-Mult** in Fig. 2 on input tuples $\{(\llbracket q^{(i)}(\cdot) \rrbracket, \llbracket h^{(i, k_{i+1})}(\cdot) \rrbracket)\}_{i \in [0,d]}$.
 - (c) \mathcal{P} opens $[m_d] - \sum_{\omega \in \{0,1\}^{k_d}} [W_d(\omega)] \cdot \chi_{\omega}(\mathbf{r}_d)$, where $\chi_{\omega}(\cdot)$ is a Lagrange basis as defined in Def. 1. \mathcal{V} checks whether it is a valid opening of $[0]$.
5. \mathcal{V} accepts if and only \mathcal{P} passes all the checks above. Otherwise, \mathcal{V} rejects.

Fig. 3. Our ZK for layered arithmetic circuits in the $\mathcal{F}_{\text{rVOLE}}^{\mathbb{F}}$ -hybrid model.

be computed from $[\mathbf{x}] := \sum_{i=1}^T \lambda_r \cdot [\mathbf{x}_i]$. We remark that there exist optimizations when considering concrete instantiations of subfield VOLE. As it is beyond the scope of paper, details of optimizations are omitted.

5 Interactive LPZK for General Arithmetic Circuits

In this section, we first describe an ILPZK proof system for proving the satisfiability of general arithmetic circuits, which satisfies all properties as indicated in Theorem 2. With this new ILPZK, we can also construct a self-contained VOLE-based ZK protocol $\Pi_{\text{ZK}g}^{\mathbb{F}}$. Finally, we prove that our protocol $\Pi_{\text{ZK}g}^{\mathbb{F}}$ UC-realizes \mathcal{F}_{ZK} in the rVOLE-hybrid model with information-theoretic malicious security.

5.1 Our ILPZK Construction

In contrast to layered circuits, gates of generic circuits may take inputs from all the previous layers. Due to this nature, it remained unclear over ten years how to adapt GKR to generic circuits without an $\mathcal{O}(d)$ overhead induced by arranging generic circuits into layered circuits. Virgo++ [37] is a recent breakthrough, extending GKR to generic circuits with linear prover time, and without $\mathcal{O}(d)$ overhead, from which we distill ideas. As usual, we first explicitly list notations used in this section here.

Notations. Let $\mathcal{C} : \mathbb{F}^{s_d} \rightarrow \mathbb{F}^{s_0}$ be a general circuit over \mathbb{F} of depth d , size S , and fan-in two. We also label each layer of \mathcal{C} from 0 to d , with 0 being the output layer and d being the input layer. Each layer $i \in [0, d)$ of \mathcal{C} contains gates that each takes one input from layer $i + 1$ and another input from previous layer j , where $j = i + 1, \dots, d$. Let \mathbf{W}_i be the outputs of gates in layer $i \in [0, d]$ and define $s_i := |\mathbf{W}_i|$. Let $\mathbf{W}_{i,j}$ be the subset of outputs of gates in layer j that connect to layer i , and define $s_{i,j} := |\mathbf{W}_{i,j}|$, for $i \in [0, d)$, $j \in [i + 1, d]$. By above definitions, $S = \sum_{i=0}^{d-1} s_i$, and $s_j \geq s_{i,j}$ for all $j \in [1, d]$, $i \in [0, d)$. W.l.o.g., we always assume $s_{i,j} = 2^{k_{i,j}}$, and $s_i = 2^{k_i}$. Since each (add/mult) gate has only two inputs, there are at most $2s_i$ gates (from previous layers) connecting to gates in layer i , i.e., $2s_i \geq \sum_{j=i+1}^d s_{i,j}$. We always assume that $k_{i,i+1}$ is the largest among $\{k_{i,i+1}, \dots, k_{i,d}\}$. We also re-define $\text{add}_{i,j}, \text{mult}_{i,j} : \{0, 1\}^{k_i+k_{i,i+1}+k_{i,j}} \rightarrow \{0, 1\}$, satisfying $\text{add}_{i,j}(z, x, y) = 1$ ($\text{mult}_{i,j}(z, x, y) = 1$) if and only if gate z is an addition (multiplication) gate in layer i (corresponds to $\mathbf{W}_i(z)$) that takes one input from gate x in layer $i + 1$ (corresponds to $\mathbf{W}_{i,i+1}(x)$) and another input from gate y in layer j (corresponds to $\mathbf{W}_{i,j}(y)$).

We view each $\mathbf{W}_i \in \mathbb{F}^{s_i}$ ($\mathbf{W}_{i,j} \in \mathbb{F}^{s_{i,j}}$) as a function $W_i : \{0, 1\}^{k_i} \rightarrow \mathbb{F}$ ($W_{i,j} : \{0, 1\}^{k_{i,j}} \rightarrow \mathbb{F}$), and denote the multi-linear extension of $W_i, W_{i,j}, \text{mult}_{i,j}, \text{add}_{i,j}$ by $\widetilde{W}_i, \widetilde{W}_{i,j}, \widetilde{\text{mult}}_{i,j}, \widetilde{\text{add}}_{i,j}$, respectively. With above definitions, it holds for all $i \in [0, d)$ that

$$\begin{aligned} \widetilde{W}_i(z) &= \sum_{j=i+1}^d \sum_{x \in \{0,1\}^{k_{i,i+1}}} \sum_{y \in \{0,1\}^{k_{i,j}}} \left(\widetilde{\text{mult}}_{i,j}(z, x, y) \widetilde{W}_{i,i+1}(x) \widetilde{W}_{i,j}(y) \right. \\ &\quad \left. + \widetilde{\text{add}}_{i,j}(z, x, y) (\widetilde{W}_{i,i+1}(x) + \widetilde{W}_{i,j}(y)) \right). \end{aligned} \tag{5}$$

Protocol Overview. The ILPZK proof for general circuits shares the same bare-bone structure with that of Sect. 4, and now Eq.(5) is the key to the layer-by-layer reduction. Observe that for layer i , performing a sum-check on Eq.(5) would induce in total $d - i + 1$ sub-statements for the previous layers $i + 1, \dots, d$ (in contrast to 2 sub-statements for layer $i + 1$ in the layered circuit setting). This in turn implies that when proceeding to layer i , there would be in total $i + 1$ sub-statements from layers $0, \dots, i - 1$ (in contrast to 2 sub-statements from layer $i - 1$ in the layered circuit setting). Therefore, a more sophisticated procedure of combining these sub-statements to one needs to be applied before sum-check.

The full construction also consists of d stages, and suppose in each stage $i \in [0, d)$, the prover \mathcal{P} and the verifier \mathcal{V} start with a commitment $[\widetilde{W}_i(\mathbf{r}_i)]$, where $\mathbf{r}_i \in \mathbb{F}^{k_i}$ is determined by \mathcal{V} . They run sum-check on Eq.(5) underneath commitments, reducing to $d - i + 1$ sub-statements for previous layers $i + 1, \dots, d$. Then, they aggregate all sub-statements about layer $i + 1$ via applying “sum-check” on another equation (will be explained later), obtaining a commitment on $\widetilde{W}_{i+1}(\mathbf{r}_{i+1})$. This allows \mathcal{P} and \mathcal{V} move to stage $i + 1$.

From One Statement to Multiple Sub-statements. Observe that directly perform “sum-check” on Eq.(5) would incur asymptotic overhead, since there are in total $k_{i,i+1} + \sum_{j=i+1}^d k_{i,j}$ variables to be summed over. To maintain a linear time prover, we rewrite Eq.(5) as in [37], by padding each y of length $k_{i,j}$ to $k_{i,i+1}$ (as we assume $k_{i,i+1}$ is the largest among $\{k_{i,i+1}, \dots, k_{i,d}\}$).

$$\begin{aligned}
 \widetilde{W}_i(z) &= \sum_{x \in \{0,1\}^{k_{i,i+1}}} \sum_{j=i+1}^d \sum_{y \in \{0,1\}^{k_{i,j}}} \left(\widetilde{\text{mult}}_{i,j}(z, x, y) \widetilde{W}_{i,i+1}(x) \widetilde{W}_{i,j}(y) \right. \\
 &\quad \left. + \widetilde{\text{add}}_{i,j}(z, x, y) (\widetilde{W}_{i,i+1}(x) + \widetilde{W}_{i,j}(y)) \right) \\
 &= \sum_{x, y \in \{0,1\}^{k_{i,i+1}}} \sum_{j=i+1}^d \left(\prod_{l=k_{i,j}+1}^{k_{i,i+1}} y_l \cdot \left(\widetilde{\text{mult}}_{i,j}(z, x, y^{(i,j)}) \widetilde{W}_{i,i+1}(x) \widetilde{W}_{i,j}(y^{(i,j)}) \right. \right. \\
 &\quad \left. \left. + \widetilde{\text{add}}_{i,j}(z, x, y^{(i,j)}) (\widetilde{W}_{i,i+1}(x) + \widetilde{W}_{i,j}(y^{(i,j)})) \right) \right), \tag{6}
 \end{aligned}$$

where each $y^{(i,j)}$ refers to the first $k_{i,j}$ bits of a $y \in \{0, 1\}^{k_{i,i+1}}$. Correctness of Eq.(6) follows from the fact that

$$\sum_{y \in \{0,1\}^{k_{i,j}}} f(y) = \sum_{y \in \{0,1\}^{k_{i,i+1}}} y^{k_{i,j}+1} \cdots y^{k_{i,i+1}} f(y^{(i,j)})$$

holds for any $f : \mathbb{F}^{k_{i,j}} \rightarrow \mathbb{F}$.

Then \mathcal{P} and \mathcal{V} apply a two-phase “sum-check” on Eq.(6) very similar to that in Sect. 4, which takes $\mathcal{O}(2^{k_{i,i+1}}) = \mathcal{O}(s_{i,i+1})$ computation. In the end, they obtain sub-statements $[m_{i+1,i+1}], [m_{i,i+1}], \dots, [m_{i,d}]$, where it is supposed to hold that $m_{i+1,i+1} = \widetilde{W}_{i+1,i+1}(\bar{\mathbf{x}}^{(i)})$ and $m_{i,j} = \widetilde{W}_{i,j}(\bar{\mathbf{y}}^{(i,j)})$, for $j \in [i + 1, d]$.

Aggregate Multiple Sub-statements to One Statement. For simplicity, we define $\mathbf{W}_{i,i} = \mathbf{W}_{i-1,i}$ and $k_{i,i} = k_{i,i+1}$. Suppose for layer i , the $i + 1$ sub-statements from above layers are $[m_{0,i}], \dots, [m_{i,i}]$, where $m_{0,i} = \widetilde{W}_{0,i}(\bar{\mathbf{y}}^{(0,i)}), \dots, m_{i-1,i} = \widetilde{W}_{i-1,i}(\bar{\mathbf{y}}^{(i-1,i)})$ and $m_{i,i} = \widetilde{W}_{i,i}(\bar{\mathbf{x}}^{(i-1)})$ all hold, and $\bar{\mathbf{y}}^{(0,i)}, \dots, \bar{\mathbf{y}}^{(i-1,i)}, \bar{\mathbf{x}}^{(i-1)}$ are challenges from \mathcal{V} in previous stages. The goal is to aggregate them to one statement for \mathbf{W}_i .

As $\mathbf{W}_{0,i}, \dots, \mathbf{W}_{i,i}$ are subsets of \mathbf{W}_i , these $[m_{0,i}], \dots, [m_{i,i}]$ can be computed from \mathbf{W}_i and the previous challenges. Intuitively, this computation can be modeled as a layered arithmetic circuit \mathcal{C}_i with private input \mathbf{W}_i and output $(m_{0,i}, \dots, m_{i,i})$, on which it suffices to apply original GKR. More concisely, observe that the evaluation of a multi-linear extension can be interpreted as simple as an inner product, e.g., $\widetilde{W}_{j,i}(\bar{\mathbf{y}}^{(j,i)}) = \sum_{\omega \in \{0,1\}^{k_{0,i}}} W_{j,i}(\omega) \cdot \chi_{\omega}(\bar{\mathbf{y}}^{(j,i)})$, $j \in [0, i)$. So \mathcal{C}_i essentially do the following things: select subsets of \mathbf{W}_i , compute expansions, and finally output the inner productions.

However, this conceptually simple approach would incur $\mathcal{O}(\log S)$ overhead in proof size, as computing expansions of $\bar{\mathbf{y}}^{(j,i)}$ requires circuits of depth $\mathcal{O}(k_{j,i})$. In fact, there exists a more efficient solution by fully exploiting the summation structure involved in the inner-product. Define $\text{EQ}_{j,i} : \{0, 1\}^{k_i} \times \{0, 1\}^{k_{j,i}} \rightarrow \mathbb{F}$, where $j \in [0, i)$, which takes as input a label z that indicates gates in layer i (corresponds to $\mathbf{W}_i(z)$), and a label y that indicates gates in layer i that connect to layer j (corresponds to $\mathbf{W}_{j,i}(y)$), outputs 1 if and only if they are exactly the same gate. The following holds:

$$\begin{aligned} m_{j,i} &= \widetilde{W}_{j,i}(\bar{\mathbf{y}}^{(j,i)}) = \sum_{\omega \in \{0,1\}^{k_{j,i}}} \widetilde{W}_{j,i}(\omega) \cdot \chi_{\omega}(\bar{\mathbf{y}}^{(j,i)}) \\ &= \sum_{\omega \in \{0,1\}^{k_{j,i}}} \sum_{z \in \{0,1\}^{k_i}} \widetilde{W}_i(z) \cdot \text{EQ}_{j,i}(z, \omega) \cdot \chi_{\omega}(\bar{\mathbf{y}}^{(j,i)}) \\ &= \sum_{z \in \{0,1\}^{k_i}} \widetilde{W}_i(z) \cdot \widetilde{\text{EQ}}_{j,i}(z, \bar{\mathbf{y}}^{(j,i)}). \end{aligned} \tag{7}$$

This allows to combine the sub-statements by taking a random linear combination on Eq.(7). Let \mathcal{V} samples $\alpha^{(0,i)}, \dots, \alpha^{(i,i)} \stackrel{\$}{\leftarrow} \mathbb{F}$, we have the following.

$$\begin{aligned} \underbrace{\sum_{j=0}^i \alpha^{(j,i)} m_{j,i}}_{m^{(i)}} &= \sum_{z \in \{0,1\}^{k_i}} \left(\alpha^{(i,i)} \widetilde{W}_i(z) \widetilde{\text{EQ}}_{i-1,i}(z, \bar{\mathbf{x}}^{(i-1)}) + \sum_{j=0}^{i-1} \alpha^{(j,i)} \widetilde{W}_i(z) \widetilde{\text{EQ}}_{j,i}(z, \bar{\mathbf{y}}^{(j,i)}) \right) \\ &= \sum_{z \in \{0,1\}^{k_i}} \widetilde{W}_i(z) \cdot \underbrace{\left(\alpha^{(i,i)} \widetilde{\text{EQ}}_{i-1,i}(z, \bar{\mathbf{x}}^{(i-1)}) + \sum_{j=0}^{i-1} \alpha^{(j,i)} \widetilde{\text{EQ}}_{j,i}(z, \bar{\mathbf{y}}^{(j,i)}) \right)}_{I^{(i)}(z)} \end{aligned} \tag{8}$$

Note that $I^{(i)}(z)$ only depends on the circuit topology and randomness selected by \mathcal{V} , it can be locally computed by each party. Therefore, it suffices for \mathcal{P} and

\mathcal{V} to perform a “sum-check” on Eq.(8), and in the end, they would agree on a commitment $[m_i] := [\widetilde{W}_i(\bar{z}^{(i)})]$, where $\bar{z}^{(i)} \in \mathbb{F}^{k_i}$ is selected by \mathcal{V} .

5.2 Complexity

Here we analyse the complexity of our ILPZK construction for general circuits.

Round Complexity. Our construction can be divided into d stages, with each stage $i \in [1, d-2]$ having $2k_{i,i+1} + 1 + k_{i+1}$ rounds, stage 0 having $2k_{0,1} + 2 + k_1$ rounds, and stage $d-1$ having $2k_{d-1,d} + 2 + k_d$ rounds. Thus, there are $2 + \sum_{i=0}^{d-1} (k_{i,i+1} + 1 + k_{i+1}) = \mathcal{O}(d \log S)$ rounds in total.

Proof Size. The proof size is the summation of length of sub-lines generated in each stage i , where $i \in [0, d)$. For each stage i , performing a two-phase sum-check on Eq.(6) and a sum-check on Eq.(8) incurs length of $6k_{i,i+1} + 3k_{i+1}$ in total, and committing to sub-statements incurs length of $d - i + 1$. Thus, the overall proof size is $\sum_{i=0}^{d-1} (6k_{i,i+1} + 3k_{i+1} + d - i + 1) = \mathcal{O}(n + d^2 + d \log S)$. We remark that here the $\mathcal{O}(d^2)$ term is always upper bounded by $\mathcal{O}(S)$. This is due to the fact that only if layer i connects to layer j , where $j < i$, then the sub-statement $[m_{j,i}]$ needs to be generated, yielding the number of sub-statements bounded by $2S$. This implies that only when the circuit is very narrow and almost every two layers are connected, then the proof size should be recognized as $\mathcal{O}(S)$.

Prover Time. Applying “sum-check” dominates the prover time. By Lemma 2, for each stage i , performing a two-phase sum-check for generating sub-statements costs prover time $2 \sum_{j=0}^{k_{i,i+1}} \mathcal{O}(2^{k_{i,i+1}-j}) = \mathcal{O}(2^{k_{i,i+1}}) = \mathcal{O}(s_{i,i+1})$, and performing a sum-check for combining sub-statements costs prover time $\sum_{j=0}^{k_{i+1}} \mathcal{O}(2^{k_{i+1}-j}) = \mathcal{O}(2^{k_{i+1}}) = \mathcal{O}(s_{i+1})$. Thus, it takes the prover overall $\mathcal{O}(S)$ time.

Verifier Time. By a similar argument as in Sect. 4.2, the verifier \mathcal{V} runs in time $\mathcal{O}(n + s_0 + d \log S + d^2 + T)$, where $\mathcal{O}(T)$ is the total time of evaluating $\widetilde{\text{mult}}_{i,j}$, $\widetilde{\text{add}}_{i,j}$ and $\widetilde{I}^{(i)}$. In general, \mathcal{V} runs in time $\mathcal{O}(S)$.

5.3 Security Proof in UC-Framework

We give a self-contained ZK protocol $\Pi_{\text{ZK}g}^{\mathbb{F}}$ in the random VOLE-hybrid model, which is based on our ILPZK proof in Sect. 5.1. In addition, we explicitly present two sum-check like sub-protocols, one for generating sub-statements ($\Pi_{\text{SC}1}$), and the other for combining sub-statements ($\Pi_{\text{SC}2}$). We also design two tailored procedures for batch-checking linear and multiplicative constraints for this setting. The detailed descriptions of our protocols $\Pi_{\text{ZK}g}^{\mathbb{F}}$, $\Pi_{\text{SC}1}$, $\Pi_{\text{SC}2}$, and the batch check procedure can be found in the full version [23]. In each stage i , where $i \in [0, d)$, protocol $\Pi_{\text{ZK}g}^{\mathbb{F}}$ sequentially invokes $\Pi_{\text{SC}1}$ and $\Pi_{\text{SC}2}$, proceeding from layer i to layer $i + 1$. At the end of stage $d - 1$, \mathcal{P} and \mathcal{V} perform checks on the degree-2 arithmetic constraints given by challenges from \mathcal{V} and the circuit. We remark

that $\Pi_{\mathbb{ZK}g}^{\mathbb{F}}$ can be also extended to support any field, via building upon subfield VOLE.

The following theorem guarantees the security of our protocol $\Pi_{\mathbb{ZK}g}^{\mathbb{F}}$. Also, Theorem 4 implies Theorem 2.

Theorem 4. *Our ZK protocol $\Pi_{\mathbb{ZK}g}^{\mathbb{F}}$ UC-realizes the ZK functionality $\mathcal{F}_{\mathbb{ZK}}$ in the $\mathcal{F}_{\text{VOLE}}^{\mathbb{F}}$ -hybrid model with information-theoretic malicious security. In particular, the environment \mathcal{Z} 's advantage is $\mathcal{O}(\frac{d \log S}{|\mathbb{F}|})$.*

Acknowledgements. The authors would like to thank Yuval Ishai and Yanhong Xu for many helpful discussions of this work. We are also very grateful for the insightful comments from anonymous reviewers. The work was supported in part by the National Key Research and Development (R&D) Program of China under Grant 2022YFA1004900 and in part by the National Natural Science Foundation of China under Grants 12031011, 12361141818, and 12101404.

References

1. Ames, S., Hazay, C., Ishai, Y., Venkitasubramanian, M.: Liger: lightweight sub-linear arguments without a trusted setup. In: CCS 2017, pp. 2087–2104. ACM (2017)
2. Baum, C., Braun, L., Munch-Hansen, A., Scholl, P.: Moz \mathbb{Z}_{2^k} arella: efficient vector-ole and zero-knowledge proofs over \mathbb{Z}_{2^k} . In: CRYPTO 2022. LNCS, vol. 13510, pp. 329–358. Springer (2022)
3. Baum, C., et al.: Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. In: CRYPTO 2023. LNCS, vol. 14085, pp. 581–615. Springer (2023)
4. Baum, C., Dittmer, S., Scholl, P., Wang, X.: Sok: vector ole-based zero-knowledge protocols. Des. Codes Cryptogr. **91**(11), 3527–3561 (2023)
5. Baum, C., Malozemoff, A.J., Rosen, M.B., Scholl, P.: Mac'n'cheese: zero-knowledge proofs for Boolean and arithmetic circuits with nested disjunctions. In: CRYPTO 2021. LNCS, vol. 12828, pp. 92–122. Springer (2021)
6. Ben-Sasson, E., Bentov, I., Horesh, Y., Riabzev, M.: Fast reed-solomon interactive oracle proofs of proximity. In: 45th International Colloquium on Automata, Languages, and Programming, ICALP 2018. LIPIcs, vol. 107, pp. 14:1–14:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2018). <https://doi.org/10.4230/LIPIcs.ICALP.2018.14>
7. Blum, A., Furst, M.L., Kearns, M.J., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: CRYPTO 1993. LNCS, vol. 773, pp. 278–291. Springer (1993)
8. Bootle, J., Cerulli, A., Ghadafi, E., Groth, J., Hajiabadi, M., Jakobsen, S.K.: Linear-time zero-knowledge proofs for arithmetic circuit satisfiability. In: ASIACRYPT 2017. LNCS, vol. 10626, pp. 336–365. Springer (2017)
9. Bootle, J., Chiesa, A., Liu, S.: Zero-knowledge iops with linear-time prover and polylogarithmic-time verifier. In: EUROCRYPT 2022. LNCS, vol. 13276, pp. 275–304. Springer (2022)
10. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: CCS 2018, pp. 896–912. ACM (2018)

11. Boyle, E., et al.: Efficient two-round OT extension and silent non-interactive secure computation. In: CCS 2019, pp. 291–308. ACM (2019)
12. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudo-random correlation generators: silent OT extension and more. In: CRYPTO 2019. LNCS, vol. 11694, pp. 489–518. Springer (2019)
13. Canetti, R.: Universally composable security: a new paradigm for cryptographic protocols. In: FOCS 2001, pp. 136–145. IEEE Computer Society (2001)
14. Cormode, G., Mitzenmacher, M., Thaler, J.: Practical verified computation with streaming interactive proofs. In: Goldwasser, S. (ed.) ITCS, 2012, pp. 90–112. ACM (2012)
15. Cramer, R., Damgård, I.: Zero-knowledge proofs for finite field arithmetic; or: can zero-knowledge be for free? In: Krawczyk, H. (ed.) CRYPTO '98. vol. 1462, pp. 424–441. Springer (1998)
16. Dittmer, S., Ishai, Y., Lu, S., Ostrovsky, R.: Improving line-point zero knowledge: two multiplications for the price of one. In: CCS 2022, pp. 829–841. ACM (2022)
17. Dittmer, S., Ishai, Y., Ostrovsky, R.: Line-point zero knowledge and its applications. In: ITC 2021. LIPIcs, vol. 199, pp. 5:1–5:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021)
18. Druk, E., Ishai, Y.: Linear-time encodable codes meeting the gilbert-varshamov bound and their cryptographic applications. In: Naor, M. (ed.) Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014, pp. 169–182. ACM (2014)
19. Goldwasser, S., Kalai, Y.T., Rothblum, G.N.: Delegating computation: interactive proofs for muggles. In: Dwork, C. (ed.) STOC 2008, pp. 113–122. ACM (2008)
20. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Comput. **18**(1), 186–208 (1989)
21. Golovnev, A., Lee, J., Setty, S.T.V., Thaler, J., Wahby, R.S.: Brakedown: linear-time and field-agnostic snarks for R1CS. In: CRYPTO 2023. LNCS, vol. 14082, pp. 193–226. Springer (2023)
22. Lee, J., Setty, S., Thaler, J., Wahby, R.: Linear-time and post-quantum zero-knowledge snarks for r1cs. Cryptology ePrint Archive (2021). <https://eprint.iacr.org/2021/030>
23. Lin, F., Xing, C., Yao, Y.: Interactive line-point zero-knowledge with sublinear communication and linear computation. IACR Cryptol. ePrint Arch. 1431 (2024). <https://eprint.iacr.org/2024/1431>
24. Lin, F., Xing, C., Yao, Y.: More efficient zero-knowledge protocols over \mathbb{Z}_{2^k} via galois rings. In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IX. Lecture Notes in Computer Science, vol. 14928, pp. 424–457. Springer (2024)
25. Lund, C., Fortnow, L., Karloff, H.J., Nisan, N.: Algebraic methods for interactive proof systems. J. ACM **39**(4), 859–868 (1992)
26. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Lecture Notes in Computer Science, vol. 576, pp. 129–140. Springer (1991)
27. Roy, L.: Softspokenot: Quieter OT extension from small-field silent VOLE in the minicrypt model. In: CRYPTO 2022. LNCS, vol. 13507, pp. 657–687. Springer (2022)

28. Setty, S.T.V.: Spartan: Efficient and general-purpose zkSNARKs without trusted setup. In: CRYPTO 2020. LNCS, vol. 12172, pp. 704–737. Springer (2020)
29. Thaler, J.: Time-optimal interactive proofs for circuit evaluation. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. vol. 8043, pp. 71–89. Springer (2013)
30. Vu, V., Setty, S.T.V., Blumberg, A.J., Walfish, M.: A hybrid architecture for interactive verifiable computation. In: SP 2013, pp. 223–237. IEEE Computer Society (2013)
31. Wahby, R.S., Tzialla, I., Shelat, A., Thaler, J., Walfish, M.: Doubly-efficient zkSNARKs without trusted setup. In: SP 2018, pp. 926–943. IEEE Computer Society (2018)
32. Weng, C., Yang, K., Katz, J., Wang, X.: Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for Boolean and arithmetic circuits. In: IEEE Symposium on Security and Privacy 2021, pp. 1074–1091. IEEE (2021)
33. Weng, C., Yang, K., Yang, Z., Xie, X., Wang, X.: Antman: interactive zero-knowledge proofs with sublinear communication. In: CCS 2022, pp. 2901–2914. ACM (2022)
34. Xie, T., Zhang, J., Zhang, Y., Papamanthou, C., Song, D.: Libra: succinct zero-knowledge proofs with optimal prover computation. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 733–764. Springer (2019)
35. Xie, T., Zhang, Y., Song, D.: Orion: zero knowledge proof with linear prover time. In: CRYPTO 2022. LNCS, vol. 13510, pp. 299–328. Springer (2022)
36. Yang, K., Sarkar, P., Weng, C., Wang, X.: Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In: CCS 2021, pp. 2986–3001. ACM (2021)
37. Zhang, J., et al.: Doubly efficient interactive proofs for general arithmetic circuits with linear prover time. In: Kim, Y., Kim, J., Vigna, G., Shi, E. (eds.) CCS '21, pp. 159–177. ACM (2021)
38. Zhang, J., Xie, T., Zhang, Y., Song, D.: Transparent polynomial delegation and its applications to zero knowledge proof. In: SP 2020, pp. 859–876. IEEE (2020)



LogRobin++: Optimizing Proofs of Disjunctive Statements in VOLE-Based ZK

Carmit Hazay¹, David Heath², Vladimir Kolesnikov³, Muthuramakrishnan Venkitasubramaniam⁴, and Yibin Yang³

¹ Bar-Ilan University, Ramat Gan, Israel

Carmit.Hazay@biu.ac.il

² University of Illinois Urbana-Champaign, Urbana, USA

daheath@illinois.edu

³ Georgia Institute of Technology, Atlanta, USA

{kolesnikov,yyang811}@gatech.edu

⁴ Ligerio Inc., Rochester, USA

muthu@ligerio-inc.com

Abstract. In the Zero-Knowledge Proof (ZKP) of a *disjunctive statement*, \mathcal{P} and \mathcal{V} agree on B fan-in 2 circuits $\mathcal{C}_0, \dots, \mathcal{C}_{B-1}$ over a field \mathbb{F} ; each circuit has n_{in} inputs, n_{\times} multiplications, and one output. \mathcal{P} 's goal is to demonstrate the knowledge of a witness ($id \in [B]$, $\mathbf{w} \in \mathbb{F}^{n_{in}}$), s.t. $C_{id}(\mathbf{w}) = 0$ where neither \mathbf{w} nor id is revealed. Disjunctive statements are effective, for example, in implementing ZKP based on sequential execution of CPU steps.

This paper studies ZKP (of knowledge) protocols over disjunctive statements based on Vector OLE. Denoting by λ the statistical security parameter and let $\rho \triangleq \max\{\log |\mathbb{F}|, \lambda\}$, the previous state-of-the-art protocol Robin (Yang et al. CCS'23) required $(n_{in} + 3n_{\times}) \log |\mathbb{F}| + \mathcal{O}(\rho B)$ bits of communication with $\mathcal{O}(1)$ rounds, and Mac'n'Cheese (Baum et al. CRYPTO'21) required $(n_{in} + n_{\times}) \log |\mathbb{F}| + 2n_{\times}\rho + \mathcal{O}(\rho \log B)$ bits of communication with $\mathcal{O}(\log B)$ rounds, both in the VOLE-hybrid model. Our novel protocol LogRobin++ achieves the same functionality at the cost of $(n_{in} + n_{\times}) \log |\mathbb{F}| + \mathcal{O}(\rho \log B)$ bits of communication with $\mathcal{O}(1)$ rounds in the VOLE-hybrid model. Crucially, LogRobin++ takes advantage of two new techniques – (1) an $\mathcal{O}(\log B)$ -overhead approach to prove in ZK that an IT-MAC commitment vector contains a zero; and (2) the realization of VOLE-based ZK over a disjunctive statement, where \mathcal{P} commits only to \mathbf{w} and multiplication outputs of $\mathcal{C}_{id}(\mathbf{w})$ (as opposed to prior work where \mathcal{P} commits to \mathbf{w} and all three wires that are associated with each multiplication gate).

We implemented LogRobin++ over Boolean (i.e., \mathbb{F}_2) and arithmetic (i.e., $\mathbb{F}_{2^{61}-1}$) fields. In our experiments, including the cost of generating VOLE correlations, LogRobin++ achieved up to $170\times$ optimization over Robin in communication, resulting in up to $7\times$ (resp. $3\times$) wall-clock time improvements in a WAN-like (resp. LAN-like) setting.

Keywords: Zero-Knowledge · Disjunctions · VOLE-Based ZK

1 Introduction

Zero-Knowledge (ZK) Proofs (ZKPs) [26] allow a prover \mathcal{P} to convince a verifier \mathcal{V} that some statement is true without disclosing further information. ZKPs are essential in applications such as private blockchain [8], private program analysis [22, 38], private bug-bounty [31, 55], privacy-preserving machine learning [35, 49], and many more. In the past decade, ZKPs have received much attention, with schemes varying in performance, assumptions, and interactivity.

VOLE-Based ZK. One recent popular line of work builds ZKP protocols from *Vector Linear Oblivious Evaluation* (VOLE). This paradigm is known as *VOLE-based ZK*; see e.g. [3, 4, 6, 19–21, 34, 48, 51]. This thrust is facilitated by cheaply generated VOLE correlations (i.e., the random VOLE instances); see, e.g., [11–13, 29, 45, 52]. In VOLE-based ZK, once the cryptographic task of generating VOLE correlations is complete, the remaining protocol can be (and typically is) simple, information-theoretic¹, and extremely efficient.

For a ZK statement expressed as a fan-in 2 circuit \mathcal{C} over some field. Let $|\mathcal{C}|$ denote the number of gates in \mathcal{C} . VOLE-based ZK only requires cost (i.e., communication and computation of each party) of a small constant factor over $|\mathcal{C}|$ in terms of (extension) field elements and operations. Concretely, state-of-the-art VOLE-based ZK (e.g., **QuickSilver** [51]) can handle millions of (multiplication) gates per second on modest hardware and network. For this reason, VOLE-based ZK has proved useful in applications where the statement is large, e.g., privacy-preserving ML [36, 49], privacy-preserving static analysis [37–39], privacy-preserving string matching [40], privacy-preserving databases [33], etc.

We focus on VOLE-based ZK because it offers by far the shortest end-to-end proof time among all ZKP approaches (e.g., zkSNARKs, MPC-in-the-Head, etc.), allowing for unprecedented scale and complexity of proven statements, such as applications mentioned above. See more discussion in Sect. 1.2.

ZK Disjunctions. Traditionally, ZKP schemes (including those based on VOLE) express statements as circuits (e.g., [1, 21, 51]) or constraint systems (e.g., [9, 43]). In theory, these formats support arbitrary statements (including those written in a high-level language, e.g., C/C++) with polynomial overhead. On the other hand, these models discard useful program structures – particularly conditional control flow – which can be leveraged to improve efficiency. Namely, ZKP protocols that can non-trivially handle *disjunctive statements* – where one of B possible statements is proved – are highly desirable. For example, a real-world physical CPU performs a disjunction over the instruction set in each step.

In a disjunctive statement, \mathcal{P} and \mathcal{V} agree on B circuits $\mathcal{C}_0, \dots, \mathcal{C}_{B-1}$. Each of these circuits is referred to as a *branch*. \mathcal{P} wishes to prove her ability to evaluate one such branch to 0 without disclosing which branch is taken or *active*. The naïve strategy for handling such a disjunctive proof is to evaluate each branch separately, then use a subsequent *multiplexer* circuit to select the output of the active branch. This strategy results in a large circuit with more than $\sum_{i \in [B]} |\mathcal{C}_i|$

¹ Exceptions are the works [14, 50], exploiting the additively homomorphic encryption.

Table 1. The performance of our protocol **LogRobin++**, compared with the prior work, in the VOLE-hybrid model (i.e., we do not account here for the cost of preprocessing random VOLEs, see Sect. 2.5 and 2.4). We consider the disjunctive statement as $\mathcal{C}_0 \vee \dots \vee \mathcal{C}_{B-1}$ where each $\mathcal{C}_{i \in [B]}$ has n_{in} inputs, n_{\times} multiplications and one output. We remark that *all* protocols (including prior work) support any field. For better comparison, we list the performance over two classical fields – the Boolean field and a sufficiently large arithmetic field \mathbb{F} where $|\mathbb{F}| = \lambda^{\omega(1)}$. The computation is estimated by the number of (extension) field operations. $|\mathcal{C}|$ denotes the number of gates in each branch. The **gray box** indicates the term that only appears in \mathcal{P} 's computation, not \mathcal{V} 's.

Protocol	Field	Communication (Bits)	Rounds	Computation
Mac'n'Cheese [6]	Boolean	$n_{in} + n_{\times} + 2\lambda n_{\times} + \mathcal{O}(\lambda \log B)$	$\mathcal{O}(\log B)$	$\mathcal{O}(B \mathcal{C})$
	Arithmetic	$(n_{in} + 3n_{\times}) \log \mathbb{F} + \mathcal{O}(\log B \log \mathbb{F})$		
Robin [53]	Boolean	$n_{in} + 3n_{\times} + \mathcal{O}(\lambda B)$	$\mathcal{O}(1)$	$\mathcal{O}(B \mathcal{C})$
	Arithmetic	$(n_{in} + 3n_{\times}) \log \mathbb{F} + \mathcal{O}(B \log \mathbb{F})$		
LogRobin++	Boolean	$n_{in} + n_{\times} + \mathcal{O}(\lambda \log B)$	$\mathcal{O}(1)$	$\mathcal{O}(B \mathcal{C} + B \log B)$
	Arithmetic	$(n_{in} + n_{\times}) \log \mathbb{F} + \mathcal{O}(\log B \log \mathbb{F})$		

gates. This is obviously wasteful, as only the gates in the single active branch affect the output of the overall instruction.

The study of ZKP over disjunctive statements can be traced back to the work of Cramer et al. [17]. This research problem has become very popular in recent years due to the development of the Stacked Garbling technique [30] and its natural application to efficient ZKP of statements expressed as high-level programs; see e.g. [6, 23–25, 30, 53, 54]. In this line of work, the researchers investigated custom protocols for handling general-purpose disjunctive statements, where the cost scales only with the size of a single branch. Recent work [6, 53] has brought such techniques to the VOLE-based ZK setting. Combining VOLE-based ZK and disjunctive statements is natural, as disjunctions are common and useful in large and complex statements. This is the focus of our work.

1.1 Our Results

In this work, we improve the handling of disjunctive statements in the VOLE-based ZK paradigm. W.l.o.g., let the B branches (circuits over some field \mathbb{F}) be of equal size, with n_{in} input wires and n_{\times} multiplication gates. Let λ be the statistical security parameter and $\rho \triangleq \max\{\log |\mathbb{F}|, \lambda\}$. Then, the state-of-the-art protocol Robin [53] requires $(n_{in} + 3n_{\times}) \log |\mathbb{F}| + \mathcal{O}(\rho B)$ bits of communication and $\mathcal{O}(1)$ rounds in the VOLE-hybrid model.

We propose a novel protocol **LogRobin++**² that requires only $(n_{in} + n_{\times}) \log |\mathbb{F}| + \mathcal{O}(\rho \log B)$ bits of communication and $\mathcal{O}(1)$ rounds in the VOLE-

² We note that our main protocol **LogRobin++** does *not* follow the Robin's underlying paradigm or technique. We follow the Robin naming line as Robin stands for refined oblivious branching for interactive ZK [53].

hybrid model. See Table 1 for a detailed comparison with prior state-of-the-art protocols. **LogRobin++** outperforms **Robin** in communication in two aspects: (1) its communication cost incurs an additive $\mathcal{O}(\rho \log B)$ term rather than $\mathcal{O}(\rho B)$; and (2) it saves transmission of $2n_\times$ field elements, resulting in $\approx 3\times$ improvement. To achieve these two improvements, we introduce two novel techniques:

- Inspired by [27], we propose a new technique for proving in ZK that a length- B committed vector (of IT-MAC commitments used by VOLE-based ZK) contains at least one zero element. Our technique requires transmission of only $\mathcal{O}(\log B)$ (extension) field elements. It can be directly plugged into the **Robin** protocol [53] to improve its communication to $(n_{in} + 3n_\times) \log |\mathbb{F}| + \mathcal{O}(\rho \log B)$ while keeping $\mathcal{O}(1)$ rounds, in the VOLE-hybrid model. We call this intermediate stepping-stone protocol **LogRobin**.
- We develop a new way of realizing VOLE-based ZKP of disjunctive statements. Namely, we show that with \mathcal{P} committing to *only* the inputs and multiplication outputs on the active branch (using VOLE correlations), the problem of proving a disjunction reduces to the following problem of proving the existence of an *affine* correlation among a set of quadratic ones: \mathcal{P} holds B quadratic polynomials $p_{i \in [B]}(X)$, (at least) one of which has leading coefficient 0 (i.e., it is an *affine* polynomial). \mathcal{V} holds a private evaluation point Δ and obtains a commitment to each polynomial as $p_{i \in [B]}(\Delta)$. \mathcal{P} must prove in ZK to \mathcal{V} that one of p_i 's is affine. The affine-polynomial-correlation problem can be solved using VOLE correlations. Put together, this reduction leads to our second stepping-stone protocol **Robin++**, which requires $(n_{in} + n_\times) \log |\mathbb{F}| + \mathcal{O}(\rho B)$ bits of communication and $\mathcal{O}(1)$ rounds in the VOLE-hybrid model.

Our final protocol **LogRobin++**, as indicated by its name, combines the underlying techniques of **LogRobin** and **Robin++**. At a high level, we show that the technical insight underlying **LogRobin**'s optimized 0-membership proof can be adapted to solve the affine-polynomial-correlation problem exploited by **Robin++**. Combining our two technical ideas requires care; directly combining the two techniques would either require $\mathcal{O}(B)$ communication or break the ZK property. See Sect. 3 for a concise technical overview of our protocols.

We remark that our paradigm of constructing **LogRobin** can be trivially generalized beyond VOLE-based ZK. In particular, it can be instantiated based on a commit-and-prove ZK [16] where the commitment scheme is linear homomorphic (e.g., the Pedersen commitment [44]).

We implemented **LogRobin++** over Boolean (i.e., \mathbb{F}_2) and arithmetic (i.e., $\mathbb{F}_{2^{61}-1}$) fields. The experimental results closely reflect the analytic costs in Table 1, as **LogRobin++**'s (and **Robin**'s) costs contain small hidden constants in \mathcal{O} . Our costs include VOLE generation. Compared to prior state-of-the-art **Robin** [53], **LogRobin++** improves communication by up to $170\times$ for disjunctions with many small branches. In terms of end-to-end execution time, **LogRobin++** outperforms **Robin** by up to $7\times$ (resp. $3\times$) in a 10 Mbps WAN-

like network (resp. 1 Gbps LAN-like network) for a wide range of parameters. See Sect. 5 for details.

We remark that LogRobin++ is secure against a static *unbounded* adversary (i.e., it is information-theoretically secure) in the VOLE-hybrid model. Somewhat surprisingly, when considering information-theoretic ZKP protocols in the VOLE-hybrid model, the price of evaluating one of many branches is now minimal in the following sense: LogRobin++ incurs only *additive* (poly)logarithmic communication as compared to the state-of-the-art (information-theoretically secure) VOLE-based ZK [21, 51] over a single active branch. Thus, the additional cost of private branching is now similar to the $\log B$ bits that would be required for \mathcal{P} to non-privately identify the active branch index to \mathcal{V} .

1.2 Related Work

VOLE-Based ZK. With the seminal work of [11] enabling cheap generation of VOLE correlations, a productive line of work on VOLE-based ZKP protocols soon emerged [3, 4, 6, 14, 19–21, 29, 34, 48, 51]. See also [5] for a survey. VOLE-based ZK is simple, information-theoretic in the VOLE-hybrid model, and efficient. Because of its efficient scaling, VOLE-based ZK is particularly useful for applications where the statement is large.

Consider a standard fan-in 2 circuit \mathcal{C} defined over some field \mathbb{F} with n_{in} inputs, n_{\times} multiplications, and $|\mathcal{C}|$ gates in total. State-of-the-art (information-theoretically secure) VOLE-based ZK [21, 51] incurs only linear costs with small constant factors – (1) \mathcal{P} transmits $n_{in} + n_{\times}$ field elements and $\mathcal{O}(1)$ extension field elements, (2) \mathcal{V} transmits $\mathcal{O}(1)$ extension field elements, and (3) \mathcal{P} and \mathcal{V} perform $\mathcal{O}(|\mathcal{C}|)$ extension field operations.

VOLE-based ZK communication cost can be further cut in half by leveraging a Random Oracle [20], or it can be reduced to sublinear by leveraging additively homomorphic encryption [50]. However, these optimizations do not substantially improve concrete performance as compared to [21, 51].

VOLE-based ZK proofs are not succinct, with the exception of [50] and [14]; [50] achieves $\mathcal{O}(|\mathcal{C}|^{3/4})$ and [14] achieves $\mathcal{O}(|\mathcal{C}|^{1/2})$ communication. Constructing a VOLE-based ZK proof system incurring $o(|\mathcal{C}|^{1/2})$ communication remains an open problem.

ZK Disjunctions. The study of ZKP protocols for disjunctive statements can be traced back to 90s, starting with the work of Cramer et al. [17]. This problem was later revisited and refined by [30], which targeted improvements to ZKPs based on *Garbled Circuits* [32, 56]. [30] described the possibility of reusing transmitted cryptographic material of the active branch to evaluate (to garbage and privately discard) inactive branches (they call this technique “stacking”). This limits communication cost to that of the single largest branch, but it still requires computation over all branches.

Following [30], a rich line of work [3, 4, 6, 19–21, 34, 48, 51] studies “stacking” ZKP protocols in the context of various ZK techniques. Among these, [6, 53] are the most relevant here, as they similarly focus on VOLE-based ZK. Our

protocol **LogRobin++** outperforms these prior works theoretically (see Table 1) and concretely (see Sect. 5). Note, [53] also studied the *batched* disjunctions – a same disjunction is repeated. We only focus on the non-batched setting.

Proving a Committed Vector Contains 0. Our work is partially inspired by the elegant work of Groth and Kohlweiss [27]. [27] proposed a public coin special honest verifier zero-knowledge proof (i.e., a Σ -protocol) that can be used to show that a vector of cryptographic commitments (with special properties) contains a zero. [27] applies this type of proof to ring signatures and zerocoin [41]. The technique underlying our stepping-stone protocol **LogRobin** can be viewed as adapting their technique to the setting of IT-MAC commitments (see Sect. 2.4). We remark that we consider a malicious \mathcal{V} and apply this 0-membership proof over disjunctive statements. Our final protocol **LogRobin++** does *not* use a proof of 0 membership; instead, it leverages a sub-component of our **LogRobin** technique.

Other Related Work. ZKP is an *enormous* and fast-growing field of research. We make a few remarks about other works in the area.

Recent work [2] showed that by applying a so-called VOLE-in-the-Head cryptographic compiler, all ZK protocols relying *only* on VOLE – including ours – can be made non-interactive and publicly verifiable.

Outside VOLE-based ZK, succinct ZK proofs enjoy significant attention. Although this remarkable line of work enables incredibly small proofs and fast verification, it suffers from expensive computation on behalf of \mathcal{P} . This highlights a strength of VOLE-based ZK: in VOLE-based ZK, \mathcal{P} 's computation is lightweight and efficient.

2 Preliminaries

2.1 Notation

- λ is the statistical security parameter (e.g., 40 or 60).
- κ is the computation security parameter (e.g., 128 or 256).
- The prover is \mathcal{P} . We refer to \mathcal{P} by she, her, hers...
- The verifier is \mathcal{V} . We refer to \mathcal{V} by he, him, his...
- $x \triangleq y$ denotes that x is *defined* as y . $x := y$ denotes that y is *assigned* to x .
- We denote that x is uniformly drawn from a set S by $x \stackrel{\$}{\leftarrow} S$.
- We denote the set $\{0, \dots, n-1\}$ by $[n]$.
- We denote a finite field of size p by \mathbb{F}_p where $p \geq 2$ is a prime or a power of a prime. We use \mathbb{F} to represent a sufficiently large field, i.e., $|\mathbb{F}| = \lambda^{\omega(1)}$.
- We denote row vectors by bold lower-case letters (e.g., \mathbf{a}), where a_i (or $a[i]$) denotes the i -th component of \mathbf{a} (0-based).
- Let M be a matrix. $M_{i,j}$ is the element of i -th column and j -th row (0-based).
- We use i to index branches (e.g., $i \in [B]$), id to index the *active* branch. I.e., the id -th branch is the one that \mathcal{P} holds a valid witness.

2.2 Schwartz-Zippel-DeMillo-Lipton Lemma

The soundness of our protocols heavily relies on the *Schwartz-Zippel-DeMillo-Lipton* (SZDL) lemma [18, 46, 57], stated in Lemma 1.

Lemma 1 (Schwartz-Zippel-DeMillo-Lipton). *Let \mathbb{F} be a field and $p \in \mathbb{F}[x_1, \dots, x_n]$ be a (multivariate) polynomial of degree d . Suppose $|\mathbb{F}| > d$, then*

$$\Pr \left[p(\mathbf{v}) = 0 \mid \mathbf{v} \stackrel{\$}{\leftarrow} \mathbb{F}^n \right] \leq \frac{d}{|\mathbb{F}|}$$

2.3 Security Model

We formalize our protocol using the universally composable (UC) framework [15]. We use UC to prove security in the presence of a *malicious, static* adversary. For simplicity, we omit standard UC session (and sub-session) IDs.

2.4 IT-MACs

Information Theoretic Message Authentication Codes (IT-MACs) [10, 42] are two-party (here, between \mathcal{P} and \mathcal{V}) distributed correlated randomness that can be used as commitments. In IT-MACs over \mathbb{F} , \mathcal{V} holds a uniformly sampled *global* key $\Delta \stackrel{\$}{\leftarrow} \mathbb{F}$. For \mathcal{P} to commit a value $x \in \mathbb{F}$, \mathcal{V} samples a uniform *local* key $k_x \stackrel{\$}{\leftarrow} \mathbb{F}$ and \mathcal{P} will learn a MAC for x as $m_x \triangleq k_x - x\Delta$. We use $[x]_\Delta \triangleq \langle (x, m_x), k_x \rangle$ to denote the IT-MAC correlation of x . Δ will be eliminated when it is clear from the context. We recall the following useful properties of IT-MACs:

1. **Hiding:** k_x and Δ , held by \mathcal{V} , are independent of the committed value x .
2. **Binding:** \mathcal{P} can open $[x]$ by sending x and m_x , where \mathcal{V} would check if $k_x \stackrel{?}{=} x\Delta + m_x$. To *maliciously* open $[x]$ to $x' \neq x$ (i.e., to forge x), \mathcal{P} must guess Δ – an attack would succeed with only $\frac{1}{|\mathbb{F}|}$ probability.
3. **Linear Homomorphism:** IT-MACs support linear operations – addition/scalar multiplication/constant addition – without communication. That is, for any *public* constants c_0, c_1, \dots, c_n each in \mathbb{F} , \mathcal{P} and \mathcal{V} can *locally* generate $[c_0 + c_1x_1 + \dots + c_nx_n]$ from $[x_1], \dots, [x_n]$.³ In particular, we denote $[c_0 + c_1x_1 + \dots + c_nx_n] = c_0 + c_1 \cdot [x_1] + \dots + c_n \cdot [x_n]$. Note, this implies that an IT-MAC of a public constant can be generated *for free*.

2.5 VOLE Correlation

Random IT-MAC instances (over \mathbb{F}_p) can be generated by *Vector Oblivious Linear Evaluation* (VOLE) correlation functionality, formalized as $\mathcal{F}_{\text{VOLE}}^{p,1}$ in Fig. 1. This functionality has been widely studied, e.g., in [12, 13, 45, 48, 52]. In the

³ I.e., if $k_x = x\Delta + m_x$ and $k_y = y\Delta + m_y$, we have $(k_x + k_y) = (x + y)\Delta + (m_x + m_y)$. Moreover, for any constant $c \in \mathbb{F}$, \mathcal{P} can set $m_c = 0$ and \mathcal{V} can set $k_c = c\Delta$.

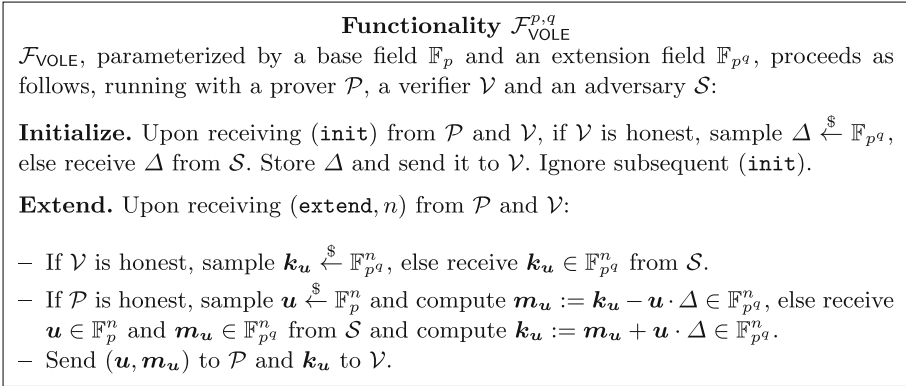


Fig. 1. The (subfield) VOLE correlation functionality.

VOLE-based ZK, \mathcal{P} and \mathcal{V} generate n instances of IT-MACs, where each IT-MAC commits an independent (pseudo-)random element $u_{j \in [n]}$. Later, it is standard [7] to *consume* one random instance $[u_j]$ to generate $[x]$ where x is chosen by \mathcal{P} . I.e., \mathcal{P} can send $x - u_j$ to allow parties to *locally* compute $[u_j] + (x - u_j) = [x]$. Note, each u_j can only be used once.

Subfield VOLE. Figure 1 also defines *subfield* VOLE correlations. This is useful when working over a small field \mathbb{F}_p . In particular, consider the Boolean field \mathbb{F}_2 . Obviously, IT-MACs over \mathbb{F}_2 do *not* provide a strong enough binding property since \mathcal{P} can successfully guess Δ with probability $\frac{1}{2}$. Naturally, we can embed values in \mathbb{F}_2 into a large enough extension field (i.e., \mathbb{F}_{2^λ}) to overcome this. However, since committed values are restricted to \mathbb{F}_2 , it is an overkill to use VOLE correlations over \mathbb{F}_{2^λ} (i.e., $\mathcal{F}_{\text{VOLE}}^{2^\lambda,1}$) to generate IT-MACs. Instead, we can exploit the subfield VOLE correlation $\mathcal{F}_{\text{VOLE}}^{2,\lambda}$ (also known as the *random correlated OT*) where each $u_{j \in [n]} \in \mathbb{F}_2 - \mathcal{P}$ sends a single bit $u_j \oplus x$ to get $[x]$.

$\mathcal{F}_{\text{VOLE}}^{p,q}$ from LPN. Recent works (e.g., [11–13, 45, 52]) show that $\mathcal{F}_{\text{VOLE}}^{p,q}$ can be instantiated efficiently via the *Learning Parity with Noise* (LPN) assumption to achieve *sublinear* costs – the **extend** instruction to generate (subfield) VOLE correlations of length n requires only $o(n)$ communications.

2.6 VOLE-Based ZK for a Single Circuit and LPZK Technique [21]

Prior work [3, 4, 6, 20, 21, 48–51] has shown that (subfield) VOLE correlations can be used as a hybrid functionality (see Fig. 1) to enable efficient ZK proofs.

Consider a circuit \mathcal{C} defined over some field \mathbb{F}_p . \mathcal{P} wishes to prove in ZK that she knows the inputs that evaluate \mathcal{C} to zero. Let q be a large enough positive integer such that $p^q = \lambda^{\omega(1)}$. VOLE-based ZK works in the commit-and-prove paradigm [16]. In particular, by exploiting functionality $\mathcal{F}_{\text{VOLE}}^{p,q}$, \mathcal{P} can commit to its inputs (i.e., the witness) and each multiplication output (i.e., the

extended witness) using IT-MACs over \mathbb{F}_{p^q} . Recall that IT-MACs are linear homomorphic. Therefore, \mathcal{P} and \mathcal{V} can *locally* evaluate \mathcal{C} over these IT-MACs. That is, the parties can put these IT-MAC commitments on \mathcal{C} 's input and each multiplication output, then evaluates \mathcal{C} gate by gate over IT-MACs. After the local evaluation, \mathcal{P} and \mathcal{V} would obtain an IT-MAC on each wire of \mathcal{C} , including the output of \mathcal{C} as $[res]$. Now, it suffices to show that each multiplication gate is formed correctly. That is, each multiplication gate connects to three wires (left input, right input and output) where each holds an IT-MAC; and \mathcal{P} needs to show that they form a multiplication triple (inside the commitments). Note, an extra multiplication needed to be added to capture the proof to show that the output of \mathcal{C} is 0, i.e., $res \cdot res = 0$ (where $[0]$ can be generated locally).

LPZK Technique. The advanced approach to proving that the multiplication relationship holds inside one IT-MAC triple is the *Line-Point Zero-Knowledge* (LPZK) technique [21, 51]. Consider $[x], [y], [z]$ where \mathcal{P} wants to prove in ZK that $z = xy$. The crucial observation is:

$$\overbrace{k_x k_y - k_z \Delta}^{\text{known by } \mathcal{V}} = (x\Delta + m_x)(y\Delta + m_y) - (z\Delta + m_z)\Delta \tag{1}$$

$$= \underbrace{(xy - z)}_{\text{known by } \mathcal{P}} \Delta^2 + \underbrace{(xm_y + ym_x - m_z)}_{\text{known by } \mathcal{P}} \Delta + \underbrace{m_x m_y}_{\text{known by } \mathcal{P}} \tag{2}$$

Hence, if $xy - z = 0$, \mathcal{P} can send two coefficients M_1 and M_0 and \mathcal{V} can check if $M_1\Delta + M_0 \stackrel{?}{=} k_x k_y - k_z \Delta$. If $xy - z \neq 0$, the equality would only hold with $\frac{2}{p}$ probability since \mathcal{P} does not know Δ . Indeed, sending $xm_y + ym_x - m_z$ breaks ZK. To recover ZK, it suffices to consume another random IT-MAC $[r]$. I.e., \mathcal{V} can compute $k_x k_y - k_z \Delta + k_r$ and \mathcal{P} can send $xm_y + ym_x - m_z + r$ and $m_x m_y + m_r$. The ZK holds since the coefficient is (uniformly) one-time padded.

Batched LPZK. Note that to prove a batch of multiplication IT-MAC triples, \mathcal{V} can issue challenges to random linearly combine coefficients induced by each triple as Equation (1). Namely, \mathcal{V} can linearly aggregate over the values known by him induced by each multiplication triple, with a \mathcal{V} -sampled public weight vector. Crucially, if each multiplication is formed correctly, \mathcal{V} should obtain a value (after the aggregation) that can be interpreted as a \mathcal{P} -known affine polynomial evaluated at Δ . On the other hand, if some multiplication does not hold, \mathcal{V} should w.h.p. obtain a value that can only be interpreted as a \mathcal{P} -known quadratic polynomial evaluated at Δ . Starting from here, the proof can be completed as the non-batched setting. We denote this procedure as the batched LPZK (check).

To further save communication, it is standard to generate the challenges (operating as the weight vector) by expanding a PRG over a κ -bit seed assuming the Random Oracle (RO) or powering an uniform field element.

By deploying the batched LPZK, the ZKP of \mathcal{C} is achieved. To summarize⁴:

⁴ We note that VOLE-based ZK works over any field.

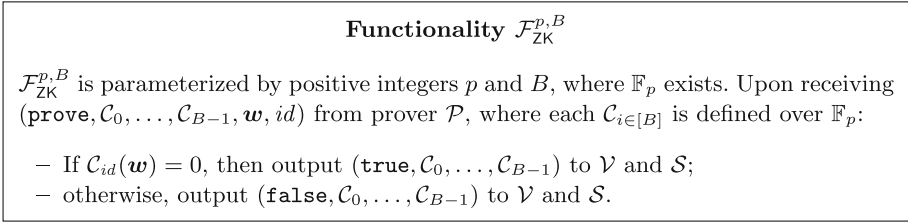


Fig. 2. The disjunctive ZK functionality.

Lemma 2 (Single-Circuit VOLE-based ZK, Informal). *For a circuit \mathcal{C} defined over \mathbb{F}_p with n_{in} inputs, n_{\times} multiplications and one output. Let $q \in \mathbb{N}$ such that $p^q = \lambda^{\omega(1)}$. There exists a constant-round ZKP protocol over \mathcal{C} with $(n_{in} + n_{\times}) \log p + 3q \log p + \mathcal{O}(1)$ bits of communication in $\mathcal{F}_{\text{VOLE}}^{p,q}$ -hybrid model.*

Remark 1. The computation complexity of VOLE-based ZK protocol of the circuit \mathcal{C} for both parties is $\mathcal{O}(|\mathcal{C}|)$ where $|\mathcal{C}|$ denotes the number of gates, in terms of field operations over \mathbb{F}_{p^q} and in the VOLE-hybrid model.

2.7 Disjunctive Statements in VOLE-Based ZK: Robin [53]

Our work focuses on studying VOLE-based ZK over *disjunctive* statements. Formally, consider B circuits $\mathcal{C}_0, \dots, \mathcal{C}_{B-1}$ defined over some field \mathbb{F}_p . \mathcal{P} 's objective is to prove to \mathcal{V} that she knows an input that evaluates (at least) 1 out of these B circuits to zero, *without revealing the identity of that branch*. We use “active branch” to denote the branch for which the prover knows a witness and let it be the id -th one. Figure 2 formalizes the disjunctive ZK functionality.

A straightforward approach to handle a disjunctive statement is to combine B circuits into one large circuit, where each circuit is included, evaluated, and finally multiplexed to determine the output. This naïve approach is undesirable as the cost would be proportional to $\mathcal{O}(B|\mathcal{C}|)$, where $|\mathcal{C}|$ denotes the maximum circuit size among *all* branches. Robin [53] shows that the communication can be optimized to be proportional to $\mathcal{O}(B + |\mathcal{C}|)$. Roughly speaking, this is achieved by reusing the “multiplication triples” of the active branch on the inactive branches.

We review Robin in slightly more detail. W.l.o.g., assume B circuits are of the same size – each has the same numbers of inputs (denoted as n_{in}) and multiplications (denoted as n_{\times}). In Robin, \mathcal{P} uses IT-MACs to commit to the n_{in} inputs (denoted as $[\mathbf{w}]$) and $3n_{\times}$ wires (denoted as $[\ell], [\mathbf{r}], [\mathbf{o}]$) associated with multiplications (left/right/output) *on the active branch*. To ensure that each multiplication is formed correctly, \mathcal{P} and \mathcal{V} perform the batched LPZK check (see Sect. 2.6). I.e., the check ensures that ℓ element-wise times \mathbf{r} is \mathbf{o} .

Then, for each branch $\mathcal{C}_{i \in [B]}$, \mathcal{P} and \mathcal{V} can evaluate \mathcal{C}_i over the committed inputs $[\mathbf{w}]$ and multiplication outputs $[\mathbf{o}]$, just like the regular VOLE-based ZK over \mathcal{C}_i (see Sect. 2.6). Note that here \mathcal{P} and \mathcal{V} reuse $[\mathbf{w}]$ and $[\mathbf{o}]$ on each branch. After evaluation, each wire on \mathcal{C}_i has an IT-MAC.

For each such branch $\mathcal{C}_i \in [B]$, denote (1) the IT-MAC vector consisting of the left wires on each multiplication as $[\ell^{(i)}]$; (2) the IT-MAC vector consisting of the right wires on each multiplication as $[\mathbf{r}^{(i)}]$; and (3) the IT-MAC on the output of \mathcal{C}_i as $[\text{res}^{(i)}]$. The crucial observation exploited by Robin is as follows: the committed $\mathbf{w}, \ell, \mathbf{r}, \mathbf{o}$ are the correct extended witness for \mathcal{C}_i if and only if the IT-MAC vector $[\ell - \ell^{(i)}] \parallel [\mathbf{r} - \mathbf{r}^{(i)}] \parallel [\text{res}^{(i)}]$ commits $0^{2n_\times + 1}$.

Therefore, to prove that \mathcal{P} indeed commits to an extended witness that satisfies one branch (conditioned on correct multiplications), it suffices to show that $0^{2n_\times + 1}$ is committed by 1-out-of- B induced IT-MAC vectors. This can be proved efficiently: by \mathcal{V} issuing a length- $(2n_\times + 1)$ random challenge vector⁵, parties can *locally* generate B IT-MACs by computing the inner product between the random challenge and each vector. Finally, it suffices to show that one of B inner products is 0 – Robin achieves this by showing that the product of these B IT-MACs is 0, which requires transmission of $\mathcal{O}(B)$ elements in \mathbb{F}_{p^q} .

Note that Robin uses the LPZK technique to prove the multiplication triples of IT-MACs in a *black-box* manner. Also note that when the circuits are defined over a small field (e.g., the Boolean field \mathbb{F}_2), the random challenge vector issued by \mathcal{V} must be defined over an extension field (e.g. \mathbb{F}_{2^λ}) to ensure soundness. We conclude this section with the following lemma and remark:

Lemma 3 (Robin, Informal). *Let $\mathcal{C}_i \in [B]$ denote B circuits (defined over \mathbb{F}_p) of the same size, where each has n_{in} inputs, n_\times multiplications and one output. Let $q \in \mathbb{N}$ such that $p^q = \lambda^{\omega(1)}$. Then, there exists a constant-round ZKP protocol for the disjunctive statement $\mathcal{C}_0 \vee \dots \vee \mathcal{C}_{B-1}$ using $(n_{in} + 3n_\times) \log p + \mathcal{O}(Bq \log p)$ bits of communication in $\mathcal{F}_{\text{VOLE}}^{p,q}$ -hybrid model.*

Remark 2. Compared to the naïve approach, the computation complexity for Robin is still $\mathcal{O}(B|\mathcal{C}|)$ in terms of number of field operations over \mathbb{F}_{p^q} .

3 Technical Overview

In this section, we provide a technical overview of our constructions. We note that understanding how Robin [53] works (see Sect. 2.7 for a concise review) would be very helpful to contextualize the components in this section.

While our protocols work over any field, for simplicity, throughout this section, consider a sufficiently large field \mathbb{F} (i.e., $|\mathbb{F}| = \lambda^{\omega(1)}$). In particular, \mathcal{P} and \mathcal{V} agree on B circuits $\mathcal{C}_i \in [B]$ defined over \mathbb{F} , each with n_{in} inputs and n_\times multiplications. Suppose \mathcal{P} wishes to prove to \mathcal{V} in ZK that she knows $\mathbf{w} \in \mathbb{F}^{n_{in}}$ that can evaluate the id -th circuit to zero. Note that id , unknown to \mathcal{V} , must be kept *private*. Moreover, let $\ell, \mathbf{r}, \mathbf{o}$ ($|\ell| = |\mathbf{r}| = |\mathbf{o}| = n_\times$) denote \mathcal{P} 's extended witness – \mathcal{P} evaluates $\mathcal{C}_{id}(\mathbf{w})$ to obtain ℓ (resp. \mathbf{r}, \mathbf{o}), which are the values on the left (resp. right, output) wire of each multiplication, in the topology order.

Roadmap. Recall that the state-of-the-art protocol Robin requires \mathcal{P} to commit to $\mathbf{w}, \ell, \mathbf{r}, \mathbf{o}$ with additive $\mathcal{O}(B)$ communication of field elements. Our final protocol LogRobin++ achieves communication costs where \mathcal{P} only needs to commit to

⁵ Again, this can be generated from a PRG or an uniform element to its powers.

\mathbf{w} and \mathbf{o} with additive $\mathcal{O}(\log B)$ communication of field elements. Our overview is presented with stepping stones and structured as follows:

1. In Sect. 3.1, we overview our first stepping stone – a technique to allow \mathcal{P} to prove to \mathcal{V} in ZK that 1-out-of- B IT-MAC commitments is 0 with $\mathcal{O}(\log B)$ communication costs. Directly plugging in this technique into Robin results in a protocol – LogRobin – that requires \mathcal{P} to commit to $\mathbf{w}, \ell, \mathbf{r}, \mathbf{o}$ with additive $\mathcal{O}(\log B)$ communication of field elements.
2. In Sect. 3.2, we overview our second stepping stone – a different way to construct VOLE-based ZK for a disjunctive statement. Essentially, we show that, by \mathcal{P} committing to only \mathbf{w} and \mathbf{o} , the proof can be reduced to show the existence of an affine correlation, where \mathcal{P} holds B *all-but-one-affine* quadratic polynomials and \mathcal{V} holds B values that are generated by evaluating these B polynomials at Δ . We construct a sub-optimal (i.e., with $\mathcal{O}(B)$ communication costs) ZK protocol to prove the existence of such an affine correlation, ultimately resulting in a protocol – Robin++ – that requires \mathcal{P} to commit to \mathbf{w}, \mathbf{o} with additive $\mathcal{O}(B)$ communication of field elements.
3. In Sect. 3.3, we overview our final protocol LogRobin++, non-trivially combining techniques underlying LogRobin and Robin++. At a very high level, we show that the technique behind proving 0 among 1-out-of- B IT-MACs (used in LogRobin) can be adapted to solve the affine-polynomial-correlation problem inside Robin++ with $\mathcal{O}(\log B)$ communication costs.

3.1 LogRobin: Optimizing the Proof of IT-MACs Containing 0

In this section, we overview the first stepping-stone protocol LogRobin. Recall that the $\mathcal{O}(B)$ communication overhead in Robin comes from \mathcal{P} proving \mathcal{V} that there is a 0 among B IT-MACs $[t_0], \dots, [t_{B-1}]$ (see Sect. 2.7). In Robin, this is done by simply multiplying the B values and opening the result to \mathcal{V} , which costs $\mathcal{O}(B)$. (If at least one multiplicand is 0, the product is 0.) The crucial technique behind LogRobin is to improve the cost of this sub-procedure to $\mathcal{O}(\log B)$.

Intuitively, this is possible as \mathcal{P} knows *where the 0 is*, while Robin only exploits the fact that the 0 exists. Informally, $\mathcal{O}(\log B)$ can be interpreted as the *minimal* amount of information required for \mathcal{P} to “point out” which element is 0 (i.e., which branch is active) *correctly and obliviously*.

A straightforward way to allow \mathcal{P} to obliviously encode which branch is active (i.e., the id -th) with $\mathcal{O}(\log B)$ overhead is to require \mathcal{P} to commit to id bit by bit (via IT-MACs). That is, w.l.o.g., let $B = 2^b$ for some $b \in \mathbb{N}$. Then, \mathcal{P} can decompose $id \in [B]$ into b bits id_0, \dots, id_{b-1} such that $id = \sum_{i=0}^{b-1} 2^i \cdot id_i$. Next, \mathcal{P} commits to each id_i as $[id_i]$ and proves in ZK that each $[id_{i \in [b]}]$ commits a bit (namely, \mathcal{P} proves that $\forall i \in [B], id_i \cdot (id_i - 1) = 0$ via the batched LPZK).

Path Matrix. Committing these bits alone is insufficient. However, it turns out that they can be exploited to further generate a powerful so-called *path matrix*, inspired by [27] (a useful technique that allows \mathcal{P} to obliviously point the active

branch). To construct the path matrix, besides $[id]$, \mathcal{P} prepares b random IT-MACs $[\delta_0], \dots, [\delta_{b-1}]$ where each $\delta_{i \in [b]} \stackrel{\$}{\leftarrow} \mathbb{F}$. Next, \mathcal{V} issues a uniform challenge $\Lambda \stackrel{\$}{\leftarrow} \mathbb{F}$. Consider the following $2 \times b$ matrix $[\mathcal{M}]$ of IT-MACs:

$$[\mathcal{M}] = \begin{pmatrix} [\Lambda \cdot (1 - id_0) + \delta_0] & [\Lambda \cdot (1 - id_1) + \delta_1] & \cdots & [\Lambda \cdot (1 - id_{b-1}) + \delta_{b-1}] \\ [\Lambda \cdot id_0 - \delta_0] & [\Lambda \cdot id_1 - \delta_1] & \cdots & [\Lambda \cdot id_{b-1} - \delta_{b-1}] \end{pmatrix}$$

The committed matrix \mathcal{M} is called the path matrix with the following properties:

- The two elements in each column differ by Λ . E.g., the two elements in the first column (within the IT-MACs) sum to $\Lambda \cdot (1 - id_0) + \delta_0 + \Lambda \cdot id_0 - \delta_0 = \Lambda$.
- Each element inside \mathcal{M} can be revealed to \mathcal{V} as $\delta_{i \in [b]}$ is uniform.
- For each column $i \in [b]$, if $id_i = 0$, the column vector \mathcal{M}_i would be $(\Lambda + \delta_i, -\delta_i)$; if $id_i = 1$, the column vector \mathcal{M}_i would be $(\delta_i, \Lambda - \delta_i)$. Essentially, Λ term *must* exist and *only* exists on the id_i -th row.

Thus, \mathcal{P} can open \mathcal{M} to \mathcal{V} without disclosing id . Note that since Λ is public, each element of $[\mathcal{M}]$ can be *locally* generated from $[id]$ and $[\delta]$. With the path matrix \mathcal{M} , the parties can bit decompose each $a \in [B]$ into a_0, \dots, a_{b-1} , then compute $\mathcal{C}_a \triangleq \prod_{i=0}^{b-1} \mathcal{M}_{i, a_i}$.

A crucial observation about each $\mathcal{C}_{a \in [B]}$ is that \mathcal{C}_a is a product of b elements involving Λ only when $a = id$. I.e., \mathcal{C}_{id} can be interpreted as a degree- b polynomial evaluated at point Λ . On the other hand, for each $a \neq id$, \mathcal{C}_a is a polynomial of degree at most $b - 1$ evaluating at Λ . The procedure to generate \mathcal{C} can be viewed as \mathcal{P} 's ability to obviously put the degree- b polynomial at \mathcal{C}_{id} .

Proving 0 exists among IT-MACs $[t_0], \dots, [t_{B-1}]$. We now present how the path matrix \mathcal{M} (in particular, the associated $\mathcal{C}_{a \in [B]}$) can be used to design a ZKP showing that $t_{id} = 0$ among $[t_{i \in [B]}]$ without disclosing id . Note that \mathcal{P} and \mathcal{V} can *locally* compute the following IT-MAC:

$$[S] \triangleq \mathcal{C}_0 \cdot [t_0] + \mathcal{C}_1 \cdot [t_1] + \cdots + \mathcal{C}_{B-1} \cdot [t_{B-1}]$$

Crucially, $\mathcal{C}_{id} \cdot [t_{id}] = [0]$ since $t_{id} = 0$. Thus, S can be interpreted as a polynomial $s(X)$ of degree at most $b - 1$, evaluated at Λ . I.e., $s(X) \triangleq \sum_{i=0}^{b-1} s_i \cdot X^i$ such that $S = s(\Lambda)$. More importantly, the coefficients s_0, \dots, s_{b-1} of $s(X)$ are known to \mathcal{P} and independent of Λ . Thus, \mathcal{P} can commit to s_0, \dots, s_{b-1} as $[s_0], \dots, [s_{b-1}]$ before Λ is sampled. Once Λ is public, \mathcal{P} proves that

$$[S] - [s_0] - \Lambda \cdot [s_1] - \cdots - \Lambda^{b-1} \cdot [s_{b-1}] = [S - s(\Lambda)]$$

commits a 0 to finish the proof. Note that the entire procedure is taken within the IT-MACs, so the ZK holds. Moreover, it only requires $\mathcal{O}(b = \log B)$ commitments, meeting our communication budget.

We briefly argue why the soundness holds. Indeed, generating the path matrix \mathcal{M} forces \mathcal{P} to select an id to claim $t_{id} = 0$. If t_0, \dots, t_{B-1} are all non-zero, $\mathcal{C}_{id} \cdot [t_{id}]$ *must* commit a degree- b polynomial evaluated at Λ . This infers that $[S]$ commits a degree- b polynomial evaluating at Λ as well. Note that Λ is uniformly chosen by \mathcal{V} and $s(X)$ is a degree- $(< b)$ polynomial chosen by \mathcal{P} before knowing Λ . Therefore, $s(\Lambda) \neq S$ w.h.p. by the SZDL lemma (see Lemma 1).

Remark 3. To prepare $s_{i \in [b]}$, \mathcal{P} needs to perform $\mathcal{O}(B \log B)$ field operations. To prepare $\mathcal{C}_{i \in [B]}$, \mathcal{P} and \mathcal{V} each only needs to perform $\mathcal{O}(B)$ field operations.

Remark 4. LogRobin is *constant-round* in the VOLE-hybrid model. While this is not our focus, this asymptotically improves over Mac'n'Cheese protocol [6].

3.2 Robin++: Committing to Lesser Values Within the Active Branch

In this section, we overview the second stepping-stone protocol Robin++. Robin++ improves over Robin by roughly $3\times$ where \mathcal{P} only needs to commit to \mathbf{w} and \mathbf{o} , whereas in Robin, \mathcal{P} commits to $\mathbf{w}, \ell, \mathbf{r}, \mathbf{o}$.

It may seem that committing to ℓ and \mathbf{r} in the disjunctive setting is inherent since it allows multiplication triples on the active branch to be reused on the inactive branch (which is the secret sauce of Robin). However, this is not the case since Robin++ only allows \mathcal{P} to commit to \mathbf{w} and \mathbf{o} . To see how Robin++ works, it is instructive to see what happens if \mathcal{P} commits to \mathbf{w} and \mathbf{o} , then \mathcal{P} and \mathcal{V} try to execute the *single-circuit* VOLE-based ZK [21, 51] (see Sect. 2.6) on each branch *reusing the committed extended witness and \mathcal{V} 's challenges*. Ensured by the soundness of the single-circuit VOLE-based ZK, the proof on the inactive branch would fail. In particular, the proofs introduce two cases for each $\mathcal{C}_{i \in [B]}$:

- **Valid (Affine):** If \mathbf{w} and \mathbf{o} are the valid extended witness of \mathcal{C}_i , based on the correctness of the single-circuit VOLE-based ZK for \mathcal{C}_i , \mathcal{P} will learn $M_1^{(i)}, M_0^{(i)} \in \mathbb{F}$ and \mathcal{V} will learn $K^{(i)} \in \mathbb{F}$ where

$$K^{(i)} = M_1^{(i)} \Delta + M_0^{(i)}$$

Recall that to finish the proof, \mathcal{P} sends two (randomized) coefficients.

- **Invalid (Quadratic):** If \mathbf{w} and \mathbf{o} are *not* the valid extended witness of \mathcal{C}_i , based on the soundness of the single-circuit VOLE-based ZK for \mathcal{C}_i , \mathcal{P} will learn $M_2^{(i)}, M_1^{(i)}, M_0^{(i)} \in \mathbb{F}$ and \mathcal{V} will learn $K^{(i)} \in \mathbb{F}$ where

$$K^{(i)} = M_2^{(i)} \Delta^2 + M_1^{(i)} \Delta + M_0^{(i)}$$

and crucially, $M_2^{(i)} \neq 0$ w.h.p. The proof fails by sending two coefficients.

Now, consider the disjunctive statement. Clearly, to show that there is an active branch, it is sufficient for \mathcal{P} to show that there is an “affine equality/correlation”. That is, instead of finishing all B proofs, \mathcal{P} and \mathcal{V} stop at the point where \mathcal{V} holds B values $K^{(i \in [B])} \in \mathbb{F}$ where each value can be interpreted as a \mathcal{P} -known quadratic polynomial evaluating at Δ (i.e., \mathcal{P} holds $p^{(i \in [B])}(X) \triangleq M_2^{(i)} X^2 + M_1^{(i)} X + M_0^{(i)}$ whereas \mathcal{V} holds $K^{(i \in [B])} \triangleq p^{(i)}(\Delta)$ and a private Δ). Starting from here, it suffices for \mathcal{P} to show in ZK that one of B evaluation points learned by \mathcal{V} is introduced by an affine polynomial. I.e., the disjunctive VOLE-based ZK proof is reduced to the above affine-polynomial-correlation problem.

A Sub-optimal Approach to Solve the Affine-Polynomial-Correlation Problem. We show a sub-optimal way to solve this problem with $\mathcal{O}(B)$ costs, resulting in Robin++. In Robin++, \mathcal{P} commits to all $M_2^{(i \in [B])}$ via IT-MACs as $\left[M_2^{(i \in [B])} \right]$ and proves in ZK to \mathcal{V} that there is a 0 among them. This step can be done using the technique used by LogRobin or just simply showing that their product is 0 as Robin. We remark that the technique used by LogRobin will *not* improve overall communication costs here since the step to commit to all $M_2^{(i \in [B])}$ costs $\mathcal{O}(B)$.

Clearly, this is insufficient – we need to further ensure that \mathcal{P} indeed commits to the correct $M_2^{(i \in [B])}$ w.r.t. each $K^{(i)}$ held by \mathcal{V} . In the non-private case without ZK, this can be done trivially by \mathcal{P} opening $M_2^{(i)}$ for each $i \in [B]$ and sending $M_1^{(i)}$ and $M_0^{(i)}$ where \mathcal{V} checks that $K^{(i)} \stackrel{?}{=} M_2^{(i)} \Delta^2 + M_1^{(i)} \Delta + M_0^{(i)}$. (Recall that Δ , sampled by \mathcal{V} , is private.) The ZK does *not* hold because (1) if $M_2^{(i)} = 0$, \mathcal{V} would know this is the active branch, and more importantly (2) $M_{1/2}^{(i)}$ are correlated with \mathcal{P} 's witness. It is classic to use fresh random IT-MACs to achieve privacy by deploying them as masks. In detail, consider two random IT-MAC instances $\left[r_1^{(i)} \right], \left[r_2^{(i)} \right]$ and the following equality:

$$\begin{aligned} \overbrace{k_{r_2}^{(i)} \Delta + k_{r_1}^{(i)}}^{\text{known by } \mathcal{V}} &= \left(r_2^{(i)} \Delta + m_{r_2}^{(i)} \right) \Delta + r_1^{(i)} \Delta + m_{r_1}^{(i)} \\ &= \underbrace{r_2^{(i)}}_{\text{known by } \mathcal{P}} \Delta^2 + \underbrace{\left(r_1^{(i)} + m_{r_2}^{(i)} \right)}_{\text{known by } \mathcal{P}} \Delta + \underbrace{m_{r_1}^{(i)}}_{\text{known by } \mathcal{P}} \end{aligned}$$

Hence, \mathcal{V} can compute $K^{(i)} + k_{r_2}^{(i)} \Delta + k_{r_1}^{(i)}$ where \mathcal{P} would open $\left[M_2^{(i)} + r_2^{(i)} \right]$ and sends $M_1^{(i)} + r_1^{(i)} + m_{r_2}^{(i)}$ and $M_0^{(i)} + m_{r_1}^{(i)}$. ZK holds now as coefficients $M_2^{(i)}$ and $M_1^{(i)}$ each is one-time-pad encrypted. In particular, \mathcal{V} would not know which branch is active since all correlations look quadratic from \mathcal{V} 's perspective. Note, QuickSilver [51] also showed a similar approach to generate and exploit this “padding” correlation, but they consume 3 random IT-MACs instead of 2.

Finally, note that the above check for each $i \in [B]$ is identical. Hence, all B checks can be performed in a batched manner. That is, \mathcal{V} issues random challenges $\chi_0, \dots, \chi_{B-1}$ and computes $\sum_{i=0}^{B-1} \chi_i K^{(i)}$ whereas \mathcal{P} computes $\sum_{i=0}^{B-1} \chi_i M_0^{(i)}$ and $\sum_{i=0}^{B-1} \chi_i M_1^{(i)}$. Furthermore, \mathcal{P} and \mathcal{V} can *locally* compute $\left[\sum_{i=0}^{B-1} \chi_i M_2^{(i)} \right]$. Random masks over the coefficients are still required to ensure the ZK property, but now only two random IT-MACs are needed in total.

To conclude, our stepping-stone protocol Robin++ exploits the reduction and the sub-optimal protocol to solve the affine-polynomial-correlation problem.

Remark 5. It is worth noting that when $B = 1$, Robin++ is (almost) identical to QuickSilver [51] – the state-of-the-art VOLE-based ZK for a single circuit. In particular, the asymptotic and concrete costs are identical.

3.3 LogRobin++: Non-trivially Combining LogRobin and Robin++

In this section, we overview our final protocol LogRobin++. As its name indicates, LogRobin++ combines the techniques exploited by Robin++ and LogRobin. With both techniques, (1) \mathcal{P} only needs to commit to \mathbf{w} and \mathbf{o} as Robin++; and (2) LogRobin++ incurs additive $\mathcal{O}(\log B)$ communication overhead as LogRobin. We remark that the combination is non-trivial as, looking ahead, a naïve attempt would either require $\mathcal{O}(B)$ costs or break the ZK property.

Recall that, by \mathcal{P} committing to only \mathbf{w} and \mathbf{o} (cf. Robin++), the disjunctive proof can be reduced to the affine-polynomial-correlation problem. I.e., \mathcal{P} and \mathcal{V} jointly hold the following correlated values:

$$\underbrace{K^{(i)}}_{\text{known by } \mathcal{V}} = \underbrace{M_2^{(i)}}_{\text{known by } \mathcal{P}} \Delta^2 + \underbrace{M_1^{(i)}}_{\text{known by } \mathcal{P}} \Delta + \underbrace{M_0^{(i)}}_{\text{known by } \mathcal{P}} \tag{3}$$

for each $i \in [B]$ (where Δ is privately sampled by \mathcal{V}), such that \mathcal{P} wishes to prove to \mathcal{V} in ZK that $\exists id \in [B], M_2^{(id)} = 0$. Robin++ achieves this by requiring \mathcal{P} to commit $M_2^{(i \in [B])}$ as $[M_2^{(0)}], \dots, [M_2^{(B-1)}]$, prove the committed B containing 0, and open a random linear combination of them (with extra uniform pads to ensure ZK). Note that committing $M_2^{(i \in [B])}$ requires $\mathcal{O}(B)$ costs!

In LogRobin++, we propose a $\mathcal{O}(\log B)$ -communication protocol to solve the affine-polynomial-correlation problem, ultimately achieving our objective.

Intuition. To get a sense of why this is possible, note that the correlation in Eq. (3) can be viewed as a “conceptual commitment” over $M_2^{(i)}$ (from \mathcal{P} to \mathcal{V}). In particular, \mathcal{P} can open the commitment via sending $M_0^{(i)}, M_1^{(i)}$ and $M_2^{(i)}$ whereas \mathcal{V} can check if Eq. (3) holds. Indeed, as the IT-MAC, if \mathcal{P} wants to forge $M_2^{(i)}$ to a different value $\widetilde{M_2^{(i)}}$, she needs to guess Δ . Viewed this way, the affine-polynomial-correlation problem can be interpreted as \mathcal{P} proving to \mathcal{V} in ZK that one of these B “conceptual commitments” is 0. Our technical insight behind LogRobin++ is to adapt our technique in LogRobin, which is used to prove 1 out of B IT-MAC commitments is 0, to these “conceptual commitments”. However, we remark that it is not ZK to open each “conceptual commitment” – the main challenge. This is because, as discussed in Sect. 3.2, $M_0^{(i)}, M_1^{(i)}$ and $M_2^{(i)}$ correlate with \mathcal{P} ’s extended witness.

Adapting LogRobin’s technique over “conceptual commitments”. Recall that \mathcal{P} in LogRobin would commit to id bit by bit, and then the parties generate a so-called path matrix \mathcal{M} . This path matrix \mathcal{M} induces B field elements $\mathcal{C}_{i \in [B]}$. By viewing each $K^{(i \in [B])}$ conceptually as a commitment, \mathcal{V} can compute

$$S \triangleq \mathcal{C}_0 K^{(0)} + \mathcal{C}_1 K^{(1)} + \dots + \mathcal{C}_{B-1} K^{(B-1)} \tag{4}$$

which can be viewed as a multivariate polynomial $s(\cdot, \cdot)$ evaluated at (Λ, Δ) as

$$S = s(\Lambda, \Delta) = \sum_{j=0}^b \sum_{k=0}^2 s_{j,k} \Lambda^j \Delta^k \tag{5}$$

where w.l.o.g., let $B = 2^b$ for some $b \in \mathbb{N}$. Note that the $3(b + 1)$ coefficients $\{s_{j,k}\}_{j \in [b+1], k \in [3]}$ are known to \mathcal{P} as they are determined by $\{M_2^{(i)}, M_1^{(i)}, M_0^{(i)}\}_{i \in [B]}$ and the \mathcal{P} -chosen id, δ (see Sect. 3.1). Recall that there is only one value within \mathcal{C} – the \mathcal{C}_{id} where id is the index of the active branch – that can be interpreted as a degree- b polynomial evaluated at Λ . Therefore, the coefficient $s_{b,2}$ of $\Lambda^b \Delta^2$ can only be induced by $\mathcal{C}_{id} K^{(id)}$ and, if \mathcal{P} is honest, *must* be 0 as $M_2^{(id)} = 0$. In other words, for $i \neq id$, since \mathcal{C}_i can only be interpreted as a degree- $< b$ polynomial evaluated at Λ , it is impossible to induce the term $\Lambda^b \Delta^2$.

Just as LogRobin, based on the SZDL lemma, it suffices for \mathcal{P} to show her ability to compute S from a degree- $(b + 1)$ multivariate polynomial evaluated at (Λ, Δ) by specifying $3b + 2$ coefficients – all $s_{j \in [b+1], k \in [3]}$ except $s_{b,2}$, *before* Λ is issued. I.e., \mathcal{P} provides an oracle to \mathcal{V} to compute a degree- $(b + 1)$ multivariate polynomial $s(X, Y)$ at (Λ, Δ) whereas \mathcal{V} needs to ensure that S (computed by Eq. (4)) is equal to $s(\Lambda, \Delta)$. Note that revealing these coefficients to \mathcal{V} directly would break privacy since they are correlated with the \mathcal{P} 's witness.

As a failed attempt, we can try to mimic LogRobin to ask \mathcal{P} to commit to all coefficients as IT-MACs and later linearly evaluate the polynomial within the IT-MACs. This fails because Δ must be kept private to \mathcal{P} to preserve the binding property of the IT-MAC. That is, even after Λ is chosen, \mathcal{P} is still not able to operate on these committed coefficients to obtain $[s(\Lambda, \Delta)]$ without knowing Δ . In fact, S itself should not be learned by \mathcal{P} , since it is correlated with Δ .

Randomization over S. Instead, similar to Robin++, LogRobin++ exploits random IT-MACs correlations (generated from VOLE) to mask the coefficients. I.e., with masking, most of them can be directly revealed.

To see how it works, first consider the coefficients of $j = b$. I.e., the coefficients $s_{b,0}$ and $s_{b,1}$ (where $s_{b,2} = 0$ if \mathcal{P} is honest). These two coefficients are related to the following additive term in Equation (5):

$$s_{b,1} \Lambda^b \Delta + s_{b,0} \Lambda^b$$

Consider one *fresh* VOLE correlation $[r_b]$, where \mathcal{V} holds k_{r_b} and \mathcal{P} holds r_b, m_{r_b} such that $k_{r_b} = r_b \Delta + m_{r_b}$. If \mathcal{V} adds $k_{r_b} \Lambda^b = r_b \Lambda^b \Delta + m_{r_b} \Lambda^b$ into S (i.e., Eq. (4)), the (above) additive term induced by $\Lambda^b \Delta$ and Λ^b would become:

$$(s_{b,1} + r_b) \Lambda^b \Delta + (s_{b,0} + m_{r_b}) \Lambda^b \tag{6}$$

Crucially, r_b looks (pseudo-)random to \mathcal{V} . Thus, \mathcal{P} can directly send $s_{b,1} + r_b$ to \mathcal{V} . However, as we will discuss at the end of this section, $s_{b,0} + m_{r_b}$ cannot be disclosed to \mathcal{V} since this would break privacy – a malicious \mathcal{V}^* can learn the active index id by manipulating it. Instead, \mathcal{P} will commit to $s_{b,0} + m_{r_b}$ as $[s_{b,0} + m_{r_b}]$. It will become clear soon how this IT-MAC is used.

Let us proceed to consider coefficients of $j = 0, \dots, b - 1$. I.e., the coefficients $s_{j,0}, s_{j,1}, s_{j,2}$ for each $j \in [b]$. These three coefficients are related to the following additive term in Eq. (5):

$$s_{j,2} \Lambda^j \Delta^2 + s_{j,1} \Lambda^j \Delta + s_{j,0} \Lambda^j$$

Consider two *fresh* VOLE correlations $[r_{j,2}]$ and $[r_{j,1}]$ for each $j \in [b]$, where \mathcal{V} holds $k_{r_{j,2}}, k_{r_{j,1}}$ and \mathcal{P} holds $r_{j,2}, r_{j,1}, m_{r_{j,2}}, m_{r_{j,1}}$ such that $k_{r_{j,2}} = r_{j,2} \Delta + m_{r_{j,2}}$ and $k_{r_{j,1}} = r_{j,1} \Delta + m_{r_{j,1}}$. Similarly, \mathcal{V} can add the term $k_{r_{j,2}} \Lambda^j \Delta + k_{r_{j,1}} \Lambda^j = r_{j,2} \Lambda^j \Delta^2 + (m_{r_{j,2}} + r_{j,1}) \Lambda^j \Delta + m_{r_{j,1}} \Lambda^j$ into S (i.e., Eq. (4)), the (above) additive term induced by $\Lambda^j \Delta^2, \Lambda^j \Delta$ and Λ^j would become:

$$(s_{j,2} + r_{j,2}) \Lambda^j \Delta^2 + (s_{j,1} + m_{r_{j,2}} + r_{j,1}) \Lambda^j \Delta + (s_{j,0} + m_{r_{j,1}}) \Lambda^j$$

Again, \mathcal{P} can directly send $s_{j,2} + r_{j,2}$ and $s_{j,1} + m_{r_{j,2}} + r_{j,1}$ as they are one-time padded by uniform $r_{j,2}$ and $r_{j,1}$. Similarly, \mathcal{V} should not learn $s_{j,0} + m_{r_{j,1}}$ so \mathcal{P} commits to $s_{j,0} + m_{r_{j,1}}$ as $[s_{j,0} + m_{r_{j,1}}]$. (We will explain at the end of this section why this cannot be directly disclosed to \mathcal{V} .)

Informally, via sending these values (i.e., $2b + 1$ randomized coefficients and $b + 1$ IT-MACs), \mathcal{P} commits to a multivariate polynomial of degree less than $b + 2$, *before knowing* Λ . In particular, they will be used as the polynomial oracle.

We are now ready to show how these coefficients inside IT-MACs are used. Naturally, they are used to let \mathcal{V} evaluate the committed polynomial at (Λ, Δ) . Note that \mathcal{V} is missing $b + 1$ coefficients $s_{0,0} + m_{r_{0,1}}, \dots, s_{b-1,0} + m_{r_{b-1,1}}, s_{b,0} + m_{r_b}$ to evaluate the committed polynomial. However, the additive term in the committed polynomial related to these coefficients is independent of Δ (i.e., $\Delta^0 = 1$). Therefore, once Λ is public, parties can *locally* compute then open:

$$\Lambda^0 \cdot [s_{0,0} + m_{r_{0,1}}] + \dots + \Lambda^{b-1} \cdot [s_{b-1,0} + m_{r_{b-1,1}}] + \Lambda^b \cdot [s_{b,0} + m_{r_b}] \quad (7)$$

which, together with $2b + 1$ randomized coefficients, helps \mathcal{V} evaluate the committed polynomial at (Λ, Δ) . Finally, if the evaluation output equals the randomized S (cf. Eq. (4)), \mathcal{V} accepts the proof. Indeed, the above protocol overcomes the difficulty in the failed attempt as it does not require \mathcal{P} to know Δ .

We remark that the polynomial must be committed before \mathcal{P} knowing Λ , which is crucial for the soundness analysis. In particular, if \mathcal{P} is cheating with all $M_{i \in [B]}^{(2)}$ being non-zeros, S should be interpreted as a degree- $(b + 2)$ polynomial evaluated at point (Λ, Δ) , even after the randomization. Hence, it is with a negligible probability that the committed polynomial (with a degree less than $b + 2$) can evaluate to the same value at (Λ, Δ) , based on the SZDL lemma.

To conclude, the above technique solves the polynomial-affine-correlation problem with $\mathcal{O}(\log B)$ communication, ultimately resulting in **LogRobin++**.

Why Can't the Coefficients Inside the IT-MACs be Disclosed? Perhaps surprisingly, unlike other $2b + 1$ (randomized) coefficients, the $b + 1$ coefficients inside IT-MACs should *not* be directly disclosed to \mathcal{V} . Here, we justify this design choice by showing how a malicious \mathcal{V} (corrupted by \mathcal{A}) could learn the active branch

Sub-procedure Eval-IT-MAC($\mathcal{C}, [\mathbf{in}], [\mathbf{o}]$)

Eval-IT-MAC is a *local* sub-procedure executed by \mathcal{P} and \mathcal{V} . It takes (1) a circuit \mathcal{C} with n_{in} inputs, n_{\times} multiplications and 1 output; (2) an IT-MAC vector $[\mathbf{in}]$ such that $|\mathbf{in}| = n_{in}$; and (3) an IT-MAC vector $[\mathbf{o}]$ such that $|\mathbf{o}| = n_{\times}$; then produces a vector of IT-MAC triples \mathbf{t} where $|\mathbf{t}| = n_{\times} + 1$. Eval-IT-MAC proceeds as follows:

\mathcal{P} (or \mathcal{V}) sets $\mathbf{t} = \perp$, then evaluates \mathcal{C} gate-by-gate in the topology order:

1. If it is an input gate, for the j -th input, put $[in_j]$ on the wire.
2. If it is an addition gate, for the j -th addition, take the IT-MAC $[x_j]$ on the left wire and the IT-MAC $[y_j]$ on the right wire, put $[x_j] + [y_j]$ on the output wire.
3. If it is a multiplication gate, for the j -th multiplication, put $[o_j]$ on the output wire. Take the IT-MAC $[x_j]$ on the left wire and the IT-MAC $[y_j]$ on the right wire, append the IT-MAC triple $([x_j], [y_j], [o_j])$ to \mathbf{t} .

After the evaluation, take the IT-MAC $[res]$ on the output of \mathcal{C} , append the IT-MAC triple $([res], [res], [0])$ to \mathbf{t} . \mathcal{P} (or \mathcal{V}) returns \mathbf{t} .

Fig. 3. Eval-IT-MAC: The sub-procedure for parties to evaluate \mathcal{C} over IT-MACs. This sub-procedure is local since parties only perform additions over IT-MACs.

index if they were disclosed. Note that \mathcal{A} is allowed to choose global key Δ and local keys \mathbf{k} in the VOLE correlation functionality (see Fig. 1). Therefore, by \mathcal{A} setting $\Delta = 0$, each local key k equals the corresponding MAC m held by \mathcal{P} . This implies that \mathcal{A} knows each $M_0^{(i \in [B])}$ in Eq. (3). Similarly, \mathcal{A} knows m_{r_b} where $[r_b]$ is used to randomize S (see Eq. (6)). Furthermore, according to Eq. (4), $s_{b,0}$ (i.e., the coefficient of λ^b) is equal to $M_0^{(id)}$. Thus, if the coefficient $s_{b,0} + m_{r_b}$ is disclosed (see Eq. (6)), \mathcal{A} learns $s_{b,0}$. By comparing $s_{b,0}$ with each $M_0^{i \in [B]}$, \mathcal{A} can infer which $id \in [B]$ gives $M_0^{(id)} = s_{b,0}$.

4 Formalization

We UC formalize our final protocol LogRobin++. For completeness, we also formalize our stepping-stone protocols LogRobin/Robin++ in our full version [28].

4.1 Sub-procedures

In this section, we define two sub-procedures that will be used by LogRobin++ (also used by LogRobin/Robin++) as subroutines. These sub-procedures are *local*.

Eval-IT-MAC: Evaluating IT-MACs over a Circuit \mathcal{C} . The first sub-procedure allows \mathcal{P} and \mathcal{V} to evaluate a circuit \mathcal{C} on IT-MAC commitments. The sub-procedure (called Eval-IT-MAC) is formalized in Fig. 3. Clearly, the computation complexity of this sub-procedure is $\mathcal{O}(|\mathcal{C}|)$.

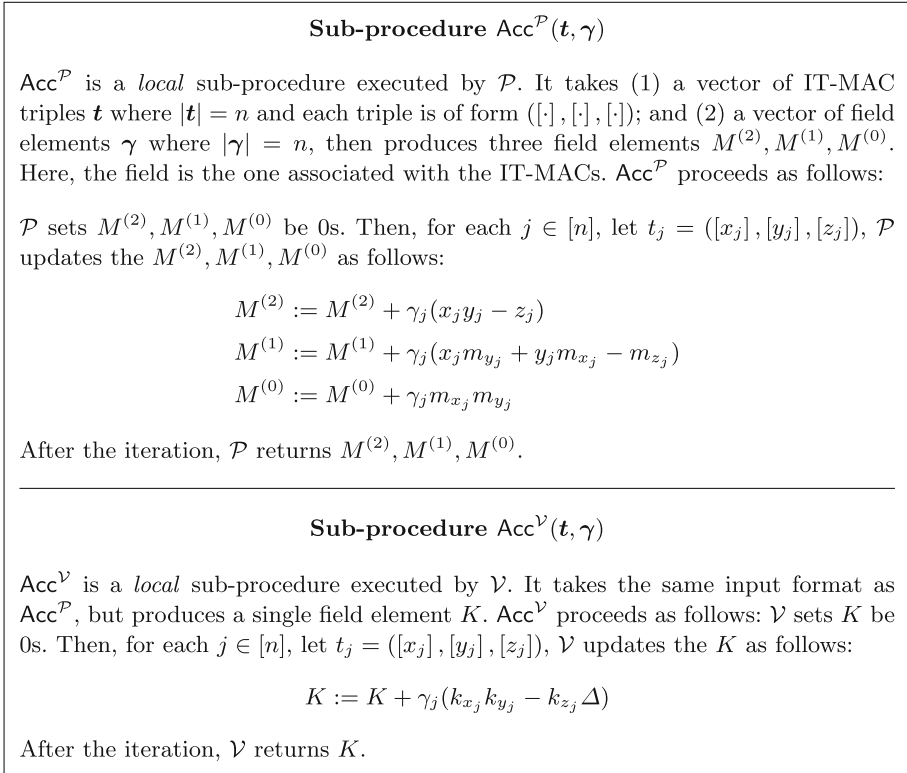


Fig. 4. $\text{Acc}^{\mathcal{P}}/\text{Acc}^{\mathcal{V}}$: The sub-procedures for \mathcal{P} and \mathcal{V} to accumulate correlations generated by IT-MAC triples. Note, if each triple in t forms a multiplication, $M^{(2)}$ is always equal to 0 *regardless* of γ .

$\text{Acc}^{\mathcal{P}}/\text{Acc}^{\mathcal{V}}$: *Linearly Accumulating IT-MAC Triples.* The second sub-procedure allows \mathcal{P} and \mathcal{V} to accumulate linearly a sequence of IT-MAC triples into a single affine or quadratic distributed correlation in Δ . This (asymmetric) sub-procedure (called $\text{Acc}^{\mathcal{P}}/\text{Acc}^{\mathcal{V}}$) is formalized in Fig. 4. This sub-procedure takes a vector of IT-MAC triples $t = (([x_j], [y_j], [z_j]))_{j \in [n]}$ where $n = |t|$ and n coefficients $\gamma_0, \dots, \gamma_{n-1}$ as inputs. Then, \mathcal{P} accumulates $M^{(2)} := \sum_{j=0}^{n-1} \gamma_j(x_j y_j - z_j)$, $M^{(1)} := \sum_{j=0}^{n-1} \gamma_j(x_j m_{y_j} + y_j m_{x_j} - m_{z_j})$, $M^{(0)} := \sum_{j=0}^{n-1} \gamma_j m_{x_j} m_{y_j}$ and \mathcal{V} accumulates $K := \sum_{j=0}^{n-1} \gamma_j(k_{x_j} k_{y_j} - k_{z_j} \Delta)$. Recall that the IT-MAC correlations ensure that $M^{(2)} \Delta^2 + M^{(1)} \Delta + M^{(0)} = K$ and, in particular, if all triples are multiplications, $M^{(2)}$ must be 0 regardless of γ . Since \mathcal{P} and \mathcal{V} perform different algorithms, we split Acc into $\text{Acc}^{\mathcal{P}}$ and $\text{Acc}^{\mathcal{V}}$, but either $\text{Acc}^{\mathcal{P}}$ or $\text{Acc}^{\mathcal{V}}$ is *local* with $\mathcal{O}(n)$ computation complexity. Our protocols will only set γ as *public coins*.

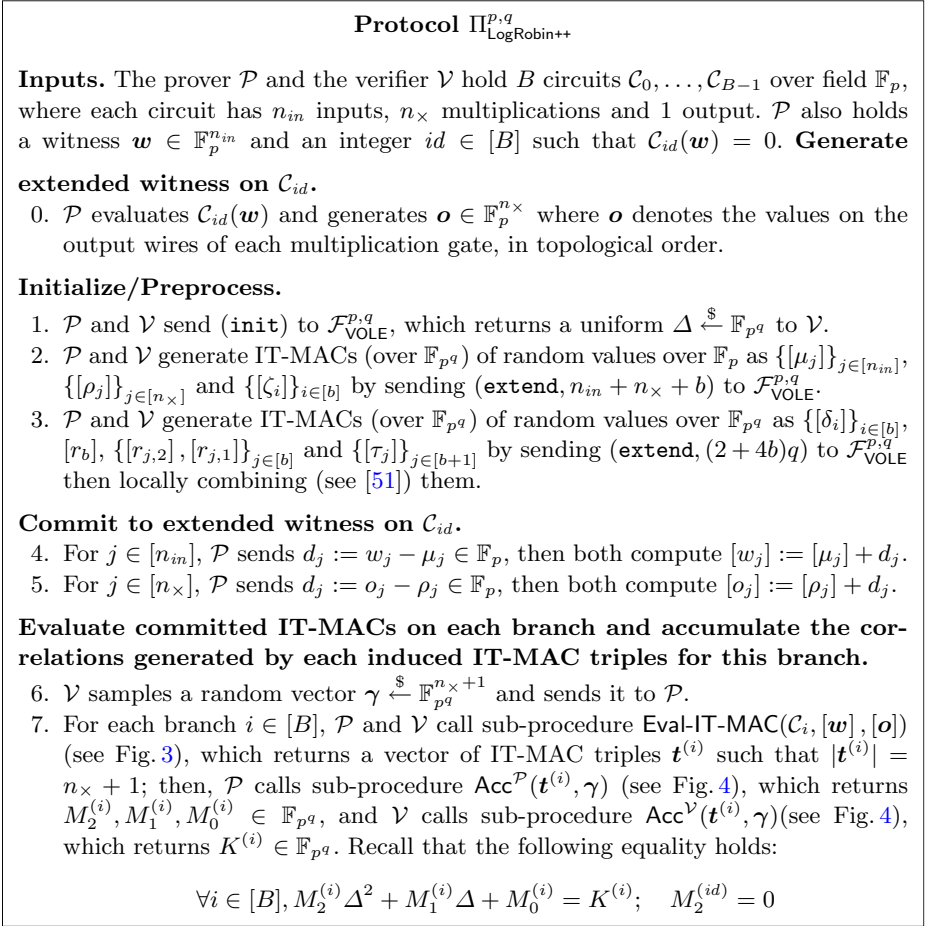


Fig. 5. LogRobin++: ZKP protocol for disjunctive circuits over any field \mathbb{F}_p in the $\mathcal{F}_{\text{VOLE}}^{p,q}$ -hybrid (see Fig. 1) model. Proceed with Fig. 6.

4.2 LogRobin++

We formalize our protocol LogRobin++ as $\Pi_{\text{LogRobin++}}^{p,q}$ in Figs. 5 and 6. We defer the reader to Sect. 3.3 for a concise technical overview of this protocol. The main security theorem associated with $\Pi_{\text{LogRobin++}}^{p,q}$ is as follows:

Theorem 1 (LogRobin++). $\Pi_{\text{LogRobin++}}^{p,q}$ (Figs. 5 and 6) UC-realizes $\mathcal{F}_{\text{ZK}}^{p,B}$ (Fig. 2) in the $\mathcal{F}_{\text{VOLE}}^{p,q}$ -hybrid model (Fig. 1) with soundness error $\frac{B+b+7}{p^q}$ (where, w.l.o.g., let $B = 2^b$ for some $b \in \mathbb{N}$) and perfect zero-knowledge, in the presence of a static unbounded adversary.

Protocol $\Pi_{\text{LogRobin++}}^{p,q}$ (Cont.)

Commit to id bit-by-bit, \mathcal{P} constructs the randomized final multivariate polynomial and declares (or commits to) its $3b + 2$ coefficients.

8. \mathcal{P} bit decomposes id as $\sum_{i=0}^{b-1} id_i \cdot 2^i$. \mathcal{P} sends $id - \zeta$ to construct $[id]$ from $[\zeta]$.
9. \mathcal{P} and \mathcal{V} execute (batched) LPZK to prove $id_i \cdot (id_i - 1) = 0$ for each $i \in [b]$.
10. \mathcal{P} constructs the following $2 \times b$ matrix, consisting of affine polynomials in X

$$\mathcal{M}(X) = \begin{pmatrix} X \cdot (1 - id_0) + \delta_0 & \cdots & X \cdot (1 - id_{b-1}) + \delta_{b-1} \\ X \cdot id_0 - \delta_0 & \cdots & X \cdot id_{b-1} - \delta_{b-1} \end{pmatrix}$$

11. \mathcal{P} constructs the multivariate polynomial in X, Y

$$s(X, Y) = \sum_{a=0}^{B-1} \left(\left(M_2^{(a)} Y^2 + M_1^{(a)} Y + M_0^{(a)} \right) \cdot \prod_{i=0}^{b-1} \mathcal{M}_{i, a_i}(X) \right)$$

where $a_{i \in [b]}$ is the bit-decomposed a , i.e., $a = \sum_{i=0}^{b-1} a_i \cdot 2^i$. Since $M_2^{(id)} = 0$, $s(X)$ is a degree- $(< b + 2)$ multivariate polynomial.

12. \mathcal{P} randomizes $s(X, Y)$: for $[r_b]$, \mathcal{P} holds r_b, m_{r_b} and computes $s(X, Y) := s(X, Y) + (r_b Y + m_{r_b}) X^b$. Then, for each $j \in [b]$, for $[r_{j,2}]$ and $[r_{j,1}]$, \mathcal{P} holds $r_{j,2}, r_{j,1}, m_{r_{j,2}}, m_{r_{j,1}}$ and computes

$$s(X, Y) := s(X, Y) + (r_{j,2} Y^2 + (r_{j,1} + m_{r_{j,2}}) Y + m_{r_{j,1}}) X^j$$

After the randomization, let $s(X, Y) = \sum_{j=0}^b \sum_{k=0}^2 s_{j,k} X^j Y^k$ where each $s_{j,k} \in \mathbb{F}_{p^q}$. In particular, if \mathcal{P} is honest, $s_{b,2} = 0$.

13. \mathcal{P} sends $s_{b,1}$ and for each $j \in [b]$, \mathcal{P} sends $s_{j,2}$ and $s_{j,1}$.
14. For each $j \in [b + 1]$, \mathcal{P} sends $d_j := s_{j,0} - \tau_j \in \mathbb{F}_{p^q}$ then parties construct $[s_{j,0}]$.

Evaluate the randomized multivariate polynomial at random point (Λ, Δ) .

15. \mathcal{V} samples a random element $\Lambda \xleftarrow{\$} \mathbb{F}_{p^q}$ and sends it to \mathcal{P} .
16. \mathcal{P} and \mathcal{V} can locally generate IT-MAC matrix $[\mathcal{M}(\Lambda)]$ from $[id]$ and $[\delta]$. Then, \mathcal{P} opens each IT-MAC in the second row of $[\mathcal{M}(\Lambda)]$, resulting \mathcal{P} and \mathcal{V} hold

$$\mathcal{M}(\Lambda) = \begin{pmatrix} \Lambda \cdot (1 - id_0) + \delta_0 & \cdots & \Lambda \cdot (1 - id_{b-1}) + \delta_{b-1} \\ \Lambda \cdot id_0 - \delta_0 & \cdots & \Lambda \cdot id_{b-1} - \delta_{b-1} \end{pmatrix} \in \mathbb{F}_{p^q}^{2 \times b}$$

17. \mathcal{V} computes

$$S := \sum_{a=0}^{B-1} \left(K^{(a)} \cdot \prod_{i=0}^{b-1} \mathcal{M}_{i, a_i}(\Lambda) \right)$$

where $a_{i \in [b]}$ is the bit-decomposed a , i.e., $a = \sum_{i=0}^b a_i \cdot 2^i$.

18. \mathcal{V} adds the randomization to S : for $[r_b]$, \mathcal{V} holds k_{r_b} and computes $S := S + k_{r_b} \Lambda^b$. Then, for each $j \in [b]$, for $[r_{j,2}]$ and $[r_{j,1}]$, \mathcal{V} holds $k_{r_{j,2}}, k_{r_{j,1}}$ and computes $S := S + (r_{j,2} \Delta + r_{j,1}) \Lambda^j$.
19. \mathcal{P} and \mathcal{V} locally construct then open the IT-MAC $[S'] = \sum_{j=0}^b \Lambda^j \cdot [s_{j,0}]$.
20. \mathcal{V} computes $S' := S' + s_{b,1} \Lambda^b \Delta$. Then, for each $j \in [b]$, \mathcal{V} computes $S' := S' + s_{j,2} \Lambda^j \Delta^2 + s_{j,1} \Lambda^j \Delta$.
21. If $S = S'$, \mathcal{V} outputs $(\text{true}, C_0, \dots, C_{B-1})$. If not (or some prior proof/open fails), \mathcal{V} outputs $(\text{false}, C_0, \dots, C_{B-1})$.

Fig. 6. LogRobin++ (Continued): ZKP protocol for disjunctive circuits over any field \mathbb{F}_p in the $\mathcal{F}_{\text{VOLE}}^{p,q}$ -hybrid (see Fig. 1) model.

Proof. The proof is performed by constructing the simulator \mathcal{S} . We need to show **completeness** (trivial, omitted); **soundness** (constructing \mathcal{S} for \mathcal{P}^*); and **Zero-Knowledge** (constructing \mathcal{S} for \mathcal{V}^*).

Zero-Knowledge, \mathcal{S} for \mathcal{V}^* : The \mathcal{S} for \mathcal{V}^* is straightforward. This is because \mathcal{V}^* receives either some elements that each is one-time padded by a uniform element (i.e., the VOLE correlation) or some elements that are determined by his transcripts (including his shares of IT-MACs and the global key Δ). That is, \mathcal{S} will interact with \mathcal{V}^* and emulate the hybrid VOLE functionality $\mathcal{F}_{\text{VOLE}}^{p,q}$ for him. Essentially, \mathcal{S} proceeds as follows:

1. For Step 1, \mathcal{S} samples the Δ for \mathcal{V}^* . Note that \mathcal{V}^* can specify his own Δ by revealing its Δ to \mathcal{S} (i.e., to the hybrid functionality $\mathcal{F}_{\text{VOLE}}^{p,q}$).
2. For Step 2 and 3, \mathcal{S} samples the local keys (i.e., the \mathcal{V}^* 's IT-MAC shares of VOLE correlations) for him. Note that \mathcal{V}^* can specify his own local keys by revealing its local keys to \mathcal{S} (i.e., to the hybrid functionality $\mathcal{F}_{\text{VOLE}}^{p,q}$).
3. For Step 4 and 5, \mathcal{S} samples and sends uniform elements in \mathbb{F}_p .
4. For Step 6, \mathcal{S} receives the challenges γ from \mathcal{V}^* .
5. For Step 7, \mathcal{S} can also execute sub-procedures Eval-IT-MAC and $\text{Acc}^{\mathcal{V}}$ (as \mathcal{V}) since it has all associated values held by \mathcal{V}^* – \mathcal{S} has $K^{(i)}$ for each $i \in [B]$.
6. For Step 8, \mathcal{S} samples and sends uniform elements in \mathbb{F}_p .
7. For Step 9, \mathcal{S} can trivially forge the ZKP by knowing Δ and all local keys. I.e., since \mathcal{S} knows all local keys and Δ , it knows what \mathcal{V}^* expects as a valid proof. Suppose this value is $\Pi \in \mathbb{F}_{p^q}$. To forge the proof, \mathcal{S} sends $C_1 \xleftarrow{\$} \mathbb{F}_{p^q}$ and $C_0 := \Pi - C_1\Delta$. (See also ZK \mathcal{S} in LPZK [21, 51].)
8. For Step 13, \mathcal{S} samples and sends uniform elements in \mathbb{F}_{p^q} . Note that, in the real-world execution, each element sent by \mathcal{P} in this step is *still* one-time padded by a uniform element in the corresponding VOLE correlation.
9. For Step 14, \mathcal{S} samples and sends uniform elements in \mathbb{F}_{p^q} .
10. For Step 15, \mathcal{S} receives the challenges Λ from \mathcal{V}^* .
11. For Step 16, \mathcal{S} opens each IT-MAC (in the second row of $[\mathcal{M}(\Lambda)]$) to a uniform sample in \mathbb{F}_{p^q} . This is possible since \mathcal{S} knows Δ and can open an IT-MAC to *any* value successfully. Now, \mathcal{S} obtains a “path matrix” $\widetilde{\mathcal{M}}$.
12. For Step 16 and 17, \mathcal{S} performs the identical computation taken by \mathcal{V} . This is possible since it has all associated values held by \mathcal{V}^* . Then, \mathcal{S} obtains S .
13. For Step 18, \mathcal{S} computes $\widetilde{S}' := S - s_{b,1}\Lambda^b\Delta - \sum_{j=0}^{b-1} (s_{j,2}\Lambda^j\Delta^2 + s_{j,1}\Lambda^j\Delta)$. Here, all s values are those sampled and sent by \mathcal{S} for Step 13. Now, \mathcal{S} opens $[S']$ to \widetilde{S}' . This possible because \mathcal{S} knows Δ . Note that what computed by \mathcal{S} is essentially the correct proof that \mathcal{V} needs to see in this step. I.e., \mathcal{V}^* would accept the proof since the equality in Step 21 *must* hold.

Indeed, the distributions seen by \mathcal{V}^* in the ideal world and the real world are *identical*. This is because \mathcal{S} replaces all one-time padded values with uniform samples (including each element in the second row of the path matrix and those coefficients sent by \mathcal{P} in Step 13) and simply determines other correlated values. The simulation is *perfect*.

Soundness, \mathcal{S} for \mathcal{P}^* : Note that \mathcal{V} in $\Pi_{\text{LogRobin++}}^{p,q}$ only sends uniform elements. Thus, \mathcal{S} , emulating $\mathcal{F}_{\text{VOLE}}^{p,q}$ for \mathcal{P}^* , can interact with \mathcal{P}^* as an honest \mathcal{V} . Since \mathcal{S} emulates $\mathcal{F}_{\text{VOLE}}^{p,q}$, it can trivially extract the (extended) witness \mathbf{w}, \mathbf{o} used by \mathcal{P}^* in Step 2 and 3. In particular, this can be done by removing the one-time pads, which are generated by $\mathcal{F}_{\text{VOLE}}^{p,q}$ and known by \mathcal{S} . Now, if the emulated honest \mathcal{V} (inside \mathcal{S}) outputs **false**, \mathcal{S} simply sends **abort** to $\mathcal{F}_{\text{ZK}}^{p,B}$, so the ideal \mathcal{V} would also output **false**. Instead, if the emulated honest \mathcal{V} (inside \mathcal{S}) outputs **true**, \mathcal{S} tries and finds $id \in [B]$ such that $\mathcal{C}_{id}(\mathbf{w}) = 0$ (if there is no such id , just set id as 0); then, \mathcal{S} sends **(prove, $\mathcal{C}_0, \dots, \mathcal{C}_{B-1}, \mathbf{w}, id$)** to $\mathcal{F}_{\text{ZK}}^{p,B}$. Finally, \mathcal{S} sends the UC environment \mathcal{E} whatever outputted by \mathcal{P}^* .

We now argue why this is a valid simulator. Note that the distributions seen by \mathcal{P}^* in the ideal world and the real world are *identical* (i.e., just some uniform challenges), so the distribution outputted by \mathcal{P}^* in the real-world execution is the same as the distribution outputted by \mathcal{S} in the ideal world. As a result, we only need to quantify the probability of the event where the ideal \mathcal{V} 's output is different from the real-world \mathcal{V} 's output. Furthermore, when the emulated honest \mathcal{V} (inside \mathcal{S}) outputs **false**, the ideal world \mathcal{V} must output **false**. Thus, we only need to quantify the probability of the event where the emulated honest \mathcal{V} outputs **true** but the ideal-world \mathcal{V} outputs **false**. Note that this happens when \mathcal{P} uses a wrong (extended) witness (in the sense that \mathbf{w} does not make any $\mathcal{C}_{i \in [B]}$ output 0) but still passes all checks. I.e., this is the soundness error.

This bad event would (only) happen in the following (chained) events:

- In Step 7, even though there exists (at least) one non-multiplication triple in each $\mathbf{t}^{(i)}$, some accumulated $M_2^{(i \in [B])}$ becomes 0. Namely, among B length- $(n_x + 1)$ vectors where none of them is all 0's, there exists (at least) 1 of them, after inner producting with the (uniformly sampled) γ in Step 6, results in 0. This would only happen with up to $\frac{B}{p^q}$ probability [53, Lemma 5.1].
- In Step 9, even though \mathcal{P}^* commits to some id_i that is not a bit, the batched LPZK does not catch it. This would only happen with up to $\frac{3}{p^q}$ probability, i.e., the soundness error of the batched LPZK technique (where the batched check is achieved via a fresh random linear combination, cf. [51]).
- In Step 16, \mathcal{P}^* forges the opening of some element(s) in $\mathcal{M}(A)$. This would only happen with up to $\frac{1}{p^q}$ based on the binding property of the IT-MAC.
- In Step 19, \mathcal{P}^* forges the opening of the IT-MAC $[S']$. This would only happen with up to $\frac{1}{p^q}$ based on the binding property of the IT-MAC.
- In Step 21, $S = S'$ (accidentally) for some sampled Λ and Δ , conditioned over all previous bad events not happening. Note that if so, (Λ, Δ) must be the root of a \mathcal{P}^* -specified (multivariate) degree- $(b + 2)$ polynomial. This is because the coefficient before $\Lambda^b \Delta^2$ must be non-zero. Thus, this would only happen with up to $\frac{b+2}{p^q}$ based on the SZDL lemma (see Lemma 1).

Hence, the union soundness error bound (i.e., the summed errors) is $\frac{B+b+7}{p^q}$.

Remark 6. Step 9 is not needed if $p = 2$ (i.e., consider Boolean circuits). This is because \mathcal{P} can only commit to bits in Step 8.

Cost Analysis. We tally the computation and communication cost of LogRobin++, in the $\mathcal{F}_{\text{VOLE}}^{p,q}$ -hybrid model (Fig. 1). The (unidirectional) communication from \mathcal{P} to \mathcal{V} consists of the following components:

1. In Step 3 and 4, \mathcal{P} sends $n_{in} + n_{\times}$ elements in \mathbb{F}_p to commit to her extended witness.
2. In Step 8, \mathcal{P} sends b elements in \mathbb{F}_p to commit to bit-decomposed id .
3. In Step 9, \mathcal{P} sends 2 elements in \mathbb{F}_{p^q} for the batched LPZK check.
4. In Step 13, \mathcal{P} sends $2b + 1$ elements in \mathbb{F}_{p^q} as coefficients.
5. In Step 14, \mathcal{P} sends $b + 1$ elements in \mathbb{F}_{p^q} to commit to coefficients.
6. In Step 16, \mathcal{P} sends $2b$ elements in \mathbb{F}_{p^q} to open the IT-MAC commitments in the second row of the path matrix.
7. In Step 19, \mathcal{P} sends 2 elements in \mathbb{F}_{p^q} to open the IT-MAC commitment.

To conclude, the overall communication from \mathcal{P} to \mathcal{V} consists of $n_{in} + n_{\times} + b$ elements in \mathbb{F}_p and $5b + 6$ elements in \mathbb{F}_{p^q} . In the other direction, the communication from \mathcal{V} to \mathcal{P} only consists of random challenges in \mathbb{F}_{p^q} . Indeed, if \mathcal{V} samples each challenge independently, this will result in sending $\Omega(n_{\times} + b)$ elements in \mathbb{F}_{p^q} . To further save the communication from \mathcal{V} to \mathcal{P} , there are the following alternative approaches to generate these challenges:

- **RO variant:** It is standard to generate each sequence of uniform challenges via expanding the PRG from a uniformly chosen κ -bit seed. This optimizes the communication from \mathcal{V} to \mathcal{P} down to $\mathcal{O}(\kappa)$. However, this variant of Robin++ requires the Random Oracle assumption. Furthermore, the soundness error would now be bounded by $\frac{B+b+7}{p^q} + \frac{Q}{2^\kappa}$, where Q denotes the number of random oracle queries made by the adversary.
- **IT variant:** We can also generate each sequence of uniform challenges via powering a single uniform element. I.e., \mathcal{V} can sample and send $\alpha \xleftarrow{\$} \mathbb{F}_{p^q}$, then parties set the challenge vector as $(1, \alpha, \alpha^2, \dots)$. Clearly, This optimizes the communication from \mathcal{V} to \mathcal{P} down to $\mathcal{O}(q \log p)$, which can be set as $\mathcal{O}(\lambda)$. While this modification preserves the information-theoretic security, the soundness error would increase because of a larger probability of creating undesirable “zeros”. E.g., in Step 7, even though a malicious \mathcal{P}^* uses an invalid extended witness that does not evaluate any branch circuit to 0, the probability that one $M_2^{(i \in [B])}$ becomes 0 would now be $\frac{Bn_{\times}}{p^q}$. (This is because a malicious \mathcal{P}^* wins the game if γ happens to be a root of one out of B degree- n_{\times} polynomials.) After adjusting these bounds, the overall soundness error would now be bounded by $\frac{Bn_{\times} + 2b + 4}{p^q}$.

For computation, clearly, \mathcal{P} 's cost is dominated by $\mathcal{O}(B|C|)$ field operations over \mathbb{F}_{p^q} in Step 7 and $\mathcal{O}(B \log B)$ field operations over \mathbb{F}_{p^q} in Step 11 to compute the coefficients of $s(X, Y)$; and \mathcal{V} 's cost is dominated by $\mathcal{O}(B|C|)$ field operations over \mathbb{F}_{p^q} in Step 7 only. Note that Step 17 only requires $\mathcal{O}(B)$ field operations.

Remark 7. The cost listed in Table 1 is based on the IT variant of LogRobin++.

5 Implementation and Benchmark

5.1 Setup

Implementation. We implemented LogRobin++ based on the open-source Robin repository [53], whose VOLE correlation functionality is implemented via the EMP Toolkit [47]. We instantiated our protocols over (1) the Boolean field \mathbb{F}_2 with $\lambda \geq 100$ and (2) the arithmetic field $\mathbb{F}_{2^{61-1}}$ with $\lambda \geq 40$, using the corresponding (subfield) VOLE functionality. For completeness, we also implemented our stepping-stone protocols LogRobin/Robin++. These simpler protocols can be useful for certain parameters.

Baseline. We use Robin [53] as our baseline. We did not compare our implementations with Mac’n’Cheese [6], as their implementation is not publicly available. However, Robin concretely outperforms Mac’n’Cheese [6]; see [53, Figure 7].

Code availability. Our implementation is publicly available at <https://github.com/gconeice/logrobinplus>.

Hardware and Network Settings. Our experiments were executed over two AWS EC2 `m5.xlarge` machines⁶ that respectively instantiated \mathcal{P} and \mathcal{V} . Each party ran single-threaded. (Our protocols can support multi-threading naturally by handling each branch in parallel; we leave research and implementation of parallelism as valuable future work.) We configured different network bandwidth settings, varying from a WAN-like 10 Mbps connection to a LAN-like 1 Gbps connection, via the Linux `tc` command.

Benchmark. We tested our implementations on statements where each branch (represented as a circuit) is chosen randomly. To reduce the physical memory needed to load all branches when B is large, we consider B *identical* randomly generated circuits. We performed experiments to show that the performance difference between executing B different circuits and B identical circuits is negligible; see Sect. 5.4. This choice of benchmark is just a proof of concept. One can always save different circuits in files and load them as needed, or programmatically generate large circuits from constant-sized descriptions as e.g. EMP Toolkit. All considered protocols only need to process each circuit once, so there is no need to load each circuit into main memory twice.

RO v.s. IT. Recall that our \mathcal{V} must flip public coins. We implemented two variants of each protocol, depending on how coin flips are handled (see discussion in Sect. 4). Coins are flipped either by (1) expanding PRGs over several κ -bit seeds chosen by \mathcal{V} , requiring a Random Oracle (RO), or (2) having \mathcal{V} uniformly sample $\mathcal{O}(1)$ elements, which is information-theoretic (IT). Our results show that the performance difference between these two variants is negligible; see Sect. 5.5. In the remainder of this section, we flip coins via RO.

⁶ Intel Xeon Platinum 8175 CPU @ 3.10 GHz, 4 vCPUs, 16 GiB Memory.

Table 2. Experiment Results with $B = 2^{22}$, $n_{in} = 10$, $n_{\times} = 100$. The time reflects the wall-clock (or end-to-end) execution time from \mathcal{P} starting the proof until \mathcal{V} accepting it. The improvements are computed as the ratio of the corresponding data between our protocols and the baseline Robin – the larger, the better.

Field	Protocol	Comm.			LAN (1 Gbps)		WAN (10 Mbps)	
		$\mathcal{P} \rightarrow \mathcal{V}$	$\mathcal{V} \rightarrow \mathcal{P}$	Total Impr.	Time(s)	Impr.	Time(s)	Impr.
\mathbb{F}_2	Robin	64 MB	28 MB	92 MB	51.2		114.1	
	LogRobin	9 KB	540 KB	549 KB	15.1	3.4×	14.8	7.7×
	Robin++	128 MB	56 MB	184 MB	94.6	0.5×	212.2	0.5×
	LogRobin++	10 KB	540 KB	550 KB	16.4	3.1×	16.1	7.1×
$\mathbb{F}_{2^{61}-1}$	Robin	32 MB	2 MB	34 MB	25.8		54.3	
	LogRobin	0.8 MB	1.7 MB	2.5 MB	27.0	1.0×	28.6	1.9×
	Robin++	64 MB	2 MB	66 MB	13.8	1.9×	68.7	0.8×
	LogRobin++	0.8 MB	1.7MB	2.5 MB	15.3	1.7×	17.3	3.1×

5.2 Overall Performance

We evaluated our approach with respect to the following parameters:

- **Benchmark “Many”:** $B = 2^{22}$, $n_{in} = 10$, $n_{\times} = 100$: Namely, there are a large number of branches, and each branch is relatively small. In this case, LogRobin++ and LogRobin should outperform Robin++ and Robin.
- **Benchmark “Large”:** $B = 2$, $n_{in} = 10$, $n_{\times} = 10^7$: Namely, there are a small number of branches, and each branch is large. In this case, LogRobin++ and Robin++ should outperform LogRobin and Robin.

Experimental Results with Many Branches. Table 2 tabulates experimental results for Benchmark “Many”. We note the following:

1. LogRobin++ (and LogRobin) achieves a significant improvement in communication cost. This improvement leads to reduced wall-clock execution time.
2. Almost all communication from \mathcal{V} to \mathcal{P} is used to generate VOLE correlations. Recall, we use the VOLE implementation from the EMP-Toolkit [47]. In their implementation, each extension generates a fixed-size ($\approx 10^7$ instances) pool of VOLE correlations [52], and in some cases, we did not exhaust the entire pool (e.g., LogRobin++ and LogRobin++ in \mathbb{F}_2 test cases). Communication from \mathcal{V} to \mathcal{P} could be fine-tuned by configuring parameters in the VOLE implementation to generate a precise number of correlations.
3. Robin++ incurs 2× overhead as compared to Robin, when operating over both \mathbb{F}_2 and $\mathbb{F}_{2^{61}-1}$. This is because n_{\times} is small. In Robin++, \mathcal{P} must commit to an additional $\approx B$ elements, and, in this benchmark, this cost supercedes Robin++’s multiplication gate improvement.
4. In our LAN setting and when considering circuits over $\mathbb{F}_{2^{61}-1}$, LogRobin did not outperform Robin in end-to-end execution time. This LAN network is fast, so communication is not the bottleneck.

Table 3. Experimental results with $B = 2, n_{in} = 10, n_{\times} = 10^7$. The time reflects the wall-clock execution time from the moment \mathcal{P} starts the proof until the moment \mathcal{V} accepts it. Improvements are computed as the ratio of the corresponding data between our protocols and the baseline Robin – larger is better.

Field	Protocol	Comm.			LAN (1 Gbps)		WAN (10 Mbps)		
		$\mathcal{P} \rightarrow \mathcal{V}$	$\mathcal{V} \rightarrow \mathcal{P}$	Total Impr.	Time(s)	Impr.	Time(s)	Impr.	
\mathbb{F}_2	Robin	3.6 MB	1.0 MB	4.6 MB	8.1		10.1		
	LogRobin	3.6 MB	1.0 MB	4.6 MB	1.0×		10.0	1.0×	
	Robin++	1.2 MB	0.5 MB	1.7 MB	2.7×	5.3	1.5×	5.9	1.7×
	LogRobin++	1.2 MB	0.5 MB	1.7 MB	2.7×	5.4	1.5×	6.1	1.7×
$\mathbb{F}_{2^{61}-1}$	Robin	230 MB	3 MB	233 MB	11.7		205.8		
	LogRobin	230 MB	3 MB	233 MB	1.0×	11.7	206.1	1.0×	
	Robin++	77 MB	1 MB	78 MB	3.0×	6.5	71.7	2.9×	
	LogRobin++	77 MB	1 MB	78 MB	3.0×	6.4	71.7	2.9×	

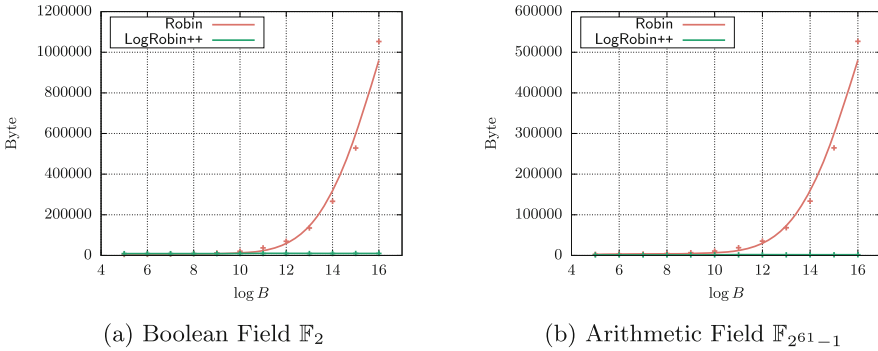


Fig. 7. Communication of LogRobin++ vs. Robin in the VOLE-hybrid Model. We fixed $n_{in} = 10, n_{\times} = 100$ and increased $b = \log B$ from 4 to 16.

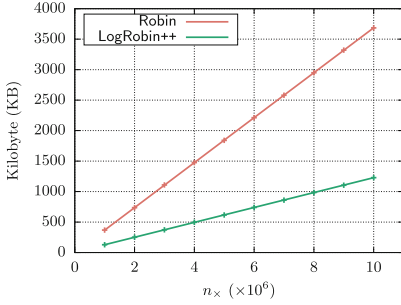
Experimental Results with Large Branches. Table 3 tabulates the experimental results for Benchmark “Large”. We note the following:

1. LogRobin++ (resp. Robin++) improved communication by 3×, reflecting our analysis.
2. In our \mathbb{F}_2 test cases, communication was relatively small. Hence, the WAN setting was not significantly slower than the LAN setting.

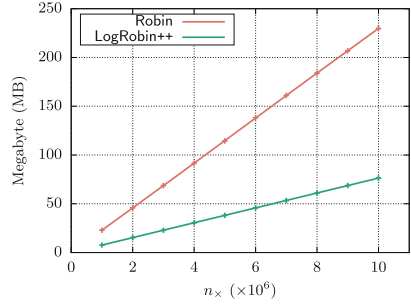
Conclusion. LogRobin++ indeed combines the improvements made by LogRobin and Robin++. Clearly, it outperforms the baseline Robin and is the best choice.

5.3 Growth Trend of Communication in the VOLE-Hybrid Model

We performed experiments to show how communication grows w.r.t. (1) increasing B , and (2) increasing $|\mathcal{C}|$. To better reflect our analysis in Sect. 4, we tested



(a) Boolean Field \mathbb{F}_2



(b) Arithmetic Field $\mathbb{F}_{2^{61}-1}$

Fig. 8. Communication of LogRobin++ vs. Robin in the VOLE-hybrid Model. We fixed $B = 2, n_{in} = 10$ and increased n_x from 1×10^6 to 10×10^6 .

Table 4. B Different Circuits v.s. B Identical Circuits in Wall-Clock Time. We set $B = 2^{10}, n_{in} = 10, n_x = 10^5$ and considered both LAN and WAN settings.

Protocol	Time (s)			
	LAN (1 Gbps)		WAN (10 Mbps)	
	Different	Identical	Different	Identical
Robin	14.1	14.6	17.0	18.3
LogRobin	13.8	13.1	17.6	17.5
Robin++	6.7	6.6	8.8	8.3
LogRobin++	6.7	7.0	8.7	8.7

and reported the communication of LogRobin++ and Robin *without* counting communication used to generate random VOLE correlations.

Communication as a Function of B. We fixed $n_{in} = 10$ and $n_x = 100$ and then tested LogRobin++ and Robin with $b = \log B$ ranging 5-16, in both the Boolean and arithmetic settings. Figure 7 plots the results. Our plots confirm that Robin’s communication grows exponentially in b while LogRobin++’s grows linearly in b .

Communication as a Function of |C|. We fixed $B = 2$ and $n_{in} = 10$ and then tested LogRobin++ and Robin with n_x ranging $1-10 \times 10^6$, in both the Boolean and arithmetic settings. Figure 8 plots the results. Our plots confirm that (1) both Robin’s and LogRobin++’s communication grows linearly in $|C|$ and (2) Robin’s communication is $\approx 3 \times$ that of LogRobin++’s.

5.4 B Identical Branches v.s. B Different Branches

We tested Robin/LogRobin/Robin++/LogRobin++ where B (randomly generated) circuits are identical or different on the arithmetic setting. The results are tabulated in Table 4. Obviously, the difference is negligible. Note that it is trivially true that the communication of these two branch configurations is the same.

Table 5. RO Variant v.s. IT Variant in Wall-Clock Time.

Parameters	Field	Protocol	Time (s)			
			LAN (1 Gbps)		WAN (10 Mbps)	
			RO	IT	RO	IT
$B = 2^{22}, n_{in} = 10, n_x = 100$	\mathbb{F}_2	Robin	51.2	49.5	114.1	115.6
		LogRobin	15.1	15.8	14.8	14.7
		Robin++	94.6	93.7	212.2	211.4
		LogRobin++	16.4	15.7	16.1	15.5
	$\mathbb{F}_{2^{61}-1}$	Robin	25.8	25.2	54.3	53.3
		LogRobin	27.0	26.9	28.6	29.1
		Robin++	13.8	13.1	68.7	69.5
		LogRobin++	15.3	15.4	17.3	17.6
$B = 2, n_{in} = 10, n_x = 10^7$	\mathbb{F}_2	Robin	8.1	8.2	10.1	10.2
		LogRobin	8.1	8.2	10.0	10.1
		Robin++	5.3	5.3	5.9	6.0
		LogRobin++	5.4	5.4	6.1	6.2
	$\mathbb{F}_{2^{61}-1}$	Robin	11.7	11.7	205.8	205.7
		LogRobin	11.7	11.6	206.1	205.8
		Robin++	6.5	6.4	71.7	71.7
		LogRobin++	6.4	6.4	71.7	71.8

5.5 RO Variant v.s. IT Variant

We tested Robin/LogRobin/Robin++/LogRobin++ each on both the RO and the IT variants. The results are tabulated in Table 5. Obviously, the difference between these two variants on each protocol is negligible. Note that it is trivially true that the communication of these two variants is the same.

Acknowledgments. This work is supported in part by Visa research award, Cisco research award, and NSF awards CNS-2246353, CNS-2246354, and CCF-2217070.

References

- Ames, S., Hazay, C., Ishai, Y., Venkatasubramanian, M.: Liger: Lightweight sub-linear arguments without a trusted setup. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 2087–2104. ACM Press, Dallas, TX, USA (Oct 31 – Nov 2, 2017). <https://doi.org/10.1145/3133956.3134104>
- Baum, C., Braun, L., Delpech de Saint Guilhem, C., Kloof, M., Orsini, E., Roy, L., Scholl, P.: Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 581–615. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38554-4_19

3. Baum, C., Braun, L., Munch-Hansen, A., Razet, B., Scholl, P.: Appenzeller to bribe: Efficient zero-knowledge proofs for mixed-mode arithmetic and \mathbb{Z}_2^k . In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 192–211. ACM Press, Virtual Event, Republic of Korea (Nov 15–19, 2021). <https://doi.org/10.1145/3460120.3484812>
4. Baum, C., Braun, L., Munch-Hansen, A., Scholl, P.: Moz \mathbb{Z}_{2^k} arella: Efficient vector-OLE and zero-knowledge proofs over \mathbb{Z}_{2^k} . In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part IV. LNCS, vol. 13510, pp. 329–358. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 15–18, 2022). https://doi.org/10.1007/978-3-031-15985-5_12
5. Baum, C., Dittmer, S., Scholl, P., Wang, X.: Sok: vector ole-based zero-knowledge protocols. *Des. Codes Cryptogr.* **91**(11), 3527–3561 (2023). <https://doi.org/10.1007/S10623-023-01292-8>
6. Baum, C., Malozemoff, A.J., Rosen, M.B., Scholl, P.: Mac’n’cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 92–122. Springer, Cham, Switzerland, Virtual Event (Aug 16–20, 2021). https://doi.org/10.1007/978-3-030-84259-8_4
7. Beaver, D.: Precomputing oblivious transfer. In: Coppersmith, D. (ed.) CRYPTO’95. LNCS, vol. 963, pp. 97–109. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 27–31, 1995). https://doi.org/10.1007/3-540-44750-4_8
8. Ben-Sasson, E., Chiesa, A., Garman, C., Green, M., Miers, I., Tromer, E., Virza, M.: Zerocash: Decentralized anonymous payments from bitcoin. In: 2014 IEEE Symposium on Security and Privacy. pp. 459–474. IEEE Computer Society Press, Berkeley, CA, USA (May 18–21, 2014). <https://doi.org/10.1109/SP.2014.36>
9. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., Virza, M.: SNARKs for C: Verifying program executions succinctly and in zero knowledge. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 90–108. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013). https://doi.org/10.1007/978-3-642-40084-1_6
10. Bendlin, R., Damgård, I., Orlandi, C., Zakarias, S.: Semi-homomorphic encryption and multiparty computation. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 169–188. Springer, Berlin, Heidelberg, Germany, Tallinn, Estonia (May 15–19, 2011). https://doi.org/10.1007/978-3-642-20465-4_11
11. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 896–912. ACM Press, Toronto, ON, Canada (Oct 15–19, 2018). <https://doi.org/10.1145/3243734.3243868>
12. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient two-round OT extension and silent non-interactive secure computation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 291–308. ACM Press, London, UK (Nov 11–15, 2019). <https://doi.org/10.1145/3319535.3354255>
13. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators: Silent OT extension and more. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 489–518. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2019). https://doi.org/10.1007/978-3-030-26954-8_16
14. Bui, D., Chu, H., Couteau, G., Wang, X., Weng, C., Yang, K., Yu, Y.: An efficient ZK compiler from SIMD circuits to general circuits. *Cryptology ePrint Archive, Report 2023/1610* (2023), <https://eprint.iacr.org/2023/1610>

15. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press, Las Vegas, NV, USA (Oct 14–17, 2001). <https://doi.org/10.1109/SFCS.2001.959888>
16. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: 34th ACM STOC. pp. 494–503. ACM Press, Montréal, Québec, Canada (May 19–21, 2002). <https://doi.org/10.1145/509907.509980>
17. Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y. (ed.) CRYPTO'94. LNCS, vol. 839, pp. 174–187. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 21–25, 1994). https://doi.org/10.1007/3-540-48658-5_19
18. DeMillo, R.A., Lipton, R.J.: A probabilistic remark on algebraic program testing. *Inf. Process. Lett.* **7**(4), 193–195 (1978)
19. Dittmer, S., Eldefrawy, K., Graham-Lengrand, S., Lu, S., Ostrovsky, R., Pereira, V.: Boosting the performance of high-assurance cryptography: Parallel execution and optimizing memory access in formally-verified line-point zero-knowledge. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 2098–2112. ACM Press, Copenhagen, Denmark (Nov 26–30, 2023). <https://doi.org/10.1145/3576915.3616583>
20. Dittmer, S., Ishai, Y., Lu, S., Ostrovsky, R.: Improving line-point zero knowledge: Two multiplications for the price of one. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 829–841. ACM Press, Los Angeles, CA, USA (Nov 7–11, 2022). <https://doi.org/10.1145/3548606.3559385>
21. Dittmer, S., Ishai, Y., Ostrovsky, R.: Line-Point Zero Knowledge and Its Applications. In: Tessaro, S. (ed.) 2nd Conference on Information-Theoretic Cryptography (ITC 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 199, pp. 5:1–5:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). <https://doi.org/10.4230/LIPIcs.ITC.2021.5>
22. Fang, Z., Darais, D., Near, J.P., Zhang, Y.: Zero knowledge static program analysis. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 2951–2967. ACM Press, Virtual Event, Republic of Korea (Nov 15–19, 2021). <https://doi.org/10.1145/3460120.3484795>
23. Goel, A., Green, M., Hall-Andersen, M., Kaptchuk, G.: Stacking sigmas: A framework to compose Σ -protocols for disjunctions. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 458–487. Springer, Cham, Switzerland, Trondheim, Norway (May 30 – Jun 3, 2022). https://doi.org/10.1007/978-3-031-07085-3_16
24. Goel, A., Hall-Andersen, M., Kaptchuk, G.: Dora: Processor expressiveness is (nearly) free in zero-knowledge for ram programs. *Cryptology ePrint Archive*, Paper 2023/1749 (2023), <https://eprint.iacr.org/2023/1749>
25. Goel, A., Hall-Andersen, M., Kaptchuk, G., Spooner, N.: Speed-stacking: Fast sublinear zero-knowledge proofs for disjunctions. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part II. LNCS, vol. 14005, pp. 347–378. Springer, Cham, Switzerland, Lyon, France (Apr 23–27, 2023). https://doi.org/10.1007/978-3-031-30617-4_12
26. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof-systems (extended abstract). In: 17th ACM STOC. pp. 291–304. ACM Press, Providence, RI, USA (May 6–8, 1985). <https://doi.org/10.1145/22145.22178>
27. Groth, J., Kohlweiss, M.: One-out-of-many proofs: Or how to leak a secret and spend a coin. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II.

- LNCS, vol. 9057, pp. 253–280. Springer, Berlin, Heidelberg, Germany, Sofia, Bulgaria (Apr 26–30, 2015). https://doi.org/10.1007/978-3-662-46803-6_9
28. Hazay, C., Heath, D., Kolesnikov, V., Venkitasubramaniam, M., Yang, Y.: LogRobin++: Optimizing proofs of disjunctive statements in VOLE-based ZK. Cryptology ePrint Archive, Paper 2024/1427 (2024), <https://eprint.iacr.org/2024/1427>
 29. Hazay, C., Yang, Y.: Toward malicious constant-rate 2PC via arithmetic garbling. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part V. LNCS, vol. 14655, pp. 401–431. Springer, Cham, Switzerland, Zurich, Switzerland (May 26–30, 2024). https://doi.org/10.1007/978-3-031-58740-5_14
 30. Heath, D., Kolesnikov, V.: Stacked garbling for disjunctive zero-knowledge proofs. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 569–598. Springer, Cham, Switzerland, Zagreb, Croatia (May 10–14, 2020). https://doi.org/10.1007/978-3-030-45727-3_19
 31. Heath, D., Yang, Y., Devecsery, D., Kolesnikov, V.: Zero knowledge for everything and everyone: Fast ZK processor with cached ORAM for ANSI C programs. In: 2021 IEEE Symposium on Security and Privacy. pp. 1538–1556. IEEE Computer Society Press, San Francisco, CA, USA (May 24–27, 2021). <https://doi.org/10.1109/SP40001.2021.00089>
 32. Jawurek, M., Kerschbaum, F., Orlandi, C.: Zero-knowledge using garbled circuits: how to prove non-algebraic statements efficiently. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013. pp. 955–966. ACM Press, Berlin, Germany (Nov 4–8, 2013). <https://doi.org/10.1145/2508859.2516662>
 33. Li, X., Weng, C., Xu, Y., Wang, X., Rogers, J.: Zksql: Verifiable and efficient query evaluation with zero-knowledge proofs. Proceedings of the VLDB Endowment **16**(8), 1804–1816 (2023)
 34. Lin, F., Xing, C., Yao, Y.: More efficient zero-knowledge protocols over \mathbb{Z}_{2^k} via galois rings. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part IX. LNCS, vol. 14928, pp. 424–457. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 18–22, 2024). https://doi.org/10.1007/978-3-031-68400-5_13
 35. Liu, T., Xie, X., Zhang, Y.: zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 2968–2985. ACM Press, Virtual Event, Republic of Korea (Nov 15–19, 2021). <https://doi.org/10.1145/3460120.3485379>
 36. Lu, T., Wang, H., Qu, W., Wang, Z., He, J., Tao, T., Chen, W., Zhang, J.: An efficient and extensible zero-knowledge proof framework for neural networks. Cryptology ePrint Archive, Paper 2024/703 (2024), <https://eprint.iacr.org/2024/703>
 37. Luick, D., Kolesar, J.C., Antonopoulos, T., Harris, W.R., Parker, J., Piskac, R., Tromer, E., Wang, X., Luo, N.: ZKSMT: A VM for proving SMT theorems in zero knowledge. In: Balzarotti, D., Xu, W. (eds.) 33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14–16, 2024. USENIX Association (2024), <https://www.usenix.org/conference/usenixsecurity24/presentation/luick>
 38. Luo, N., Antonopoulos, T., Harris, W.R., Piskac, R., Tromer, E., Wang, X.: Proving UNSAT in zero knowledge. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 2203–2217. ACM Press, Los Angeles, CA, USA (Nov 7–11, 2022). <https://doi.org/10.1145/3548606.3559373>
 39. Luo, N., Judson, S., Antonopoulos, T., Piskac, R., Wang, X.: ppSAT: Towards two-party private SAT solving. In: Butler, K.R.B., Thomas, K. (eds.) USENIX Security 2022. pp. 2983–3000. USENIX Association, Boston, MA, USA (Aug 10–12, 2022)

40. Luo, N., Weng, C., Singh, J., Tan, G., Piskac, R., Raykova, M.: Privacy-preserving regular expression matching using nondeterministic finite automata. *Cryptology ePrint Archive*, Paper 2023/643 (2023), <https://eprint.iacr.org/2023/643>
41. Miers, I., Garman, C., Green, M., Rubin, A.D.: Zerocoin: Anonymous distributed E-cash from Bitcoin. In: 2013 IEEE Symposium on Security and Privacy. pp. 397–411. IEEE Computer Society Press, Berkeley, CA, USA (May 19–22, 2013). <https://doi.org/10.1109/SP.2013.34>
42. Nielsen, J.B., Nordholt, P.S., Orlandi, C., Burra, S.S.: A new approach to practical active-secure two-party computation. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 681–700. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012). https://doi.org/10.1007/978-3-642-32009-5_40
43. Parno, B., Howell, J., Gentry, C., Raykova, M.: Pinocchio: Nearly practical verifiable computation. In: 2013 IEEE Symposium on Security and Privacy. pp. 238–252. IEEE Computer Society Press, Berkeley, CA, USA (May 19–22, 2013). <https://doi.org/10.1109/SP.2013.47>
44. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO’91. LNCS, vol. 576, pp. 129–140. Springer, Berlin, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 11–15, 1992). https://doi.org/10.1007/3-540-46766-1_9
45. Schoppmann, P., Gascón, A., Reichert, L., Raykova, M.: Distributed vector-OLE: Improved constructions and implementation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 1055–1072. ACM Press, London, UK (Nov 11–15, 2019). <https://doi.org/10.1145/3319535.3363228>
46. Schwartz, J.T.: Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)* **27**(4), 701–717 (1980)
47. Wang, X., Malozemoff, A.J., Katz, J.: EMP-toolkit: Efficient MultiParty computation toolkit. <https://github.com/emp-toolkit> (2016)
48. Weng, C., Yang, K., Katz, J., Wang, X.: Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In: 2021 IEEE Symposium on Security and Privacy. pp. 1074–1091. IEEE Computer Society Press, San Francisco, CA, USA (May 24–27, 2021). <https://doi.org/10.1109/SP40001.2021.00056>
49. Weng, C., Yang, K., Xie, X., Katz, J., Wang, X.: Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning. In: Bailey, M., Greenstadt, R. (eds.) USENIX Security 2021. pp. 501–518. USENIX Association (Aug 11–13, 2021)
50. Weng, C., Yang, K., Yang, Z., Xie, X., Wang, X.: AntMan: Interactive zero-knowledge proofs with sublinear communication. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022. pp. 2901–2914. ACM Press, Los Angeles, CA, USA (Nov 7–11, 2022). <https://doi.org/10.1145/3548606.3560667>
51. Yang, K., Sarkar, P., Weng, C., Wang, X.: QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021. pp. 2986–3001. ACM Press, Virtual Event, Republic of Korea (Nov 15–19, 2021). <https://doi.org/10.1145/3460120.3484556>
52. Yang, K., Weng, C., Lan, X., Zhang, J., Wang, X.: Ferret: Fast extension for correlated OT with small communication. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1607–1626. ACM Press, Virtual Event, USA (Nov 9–13, 2020). <https://doi.org/10.1145/3372297.3417276>

53. Yang, Y., Heath, D., Hazay, C., Kolesnikov, V., Venkatasubramanian, M.: Batchman and robin: Batched and non-batched branching for interactive ZK. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 1452–1466. ACM Press, Copenhagen, Denmark (Nov 26–30, 2023). <https://doi.org/10.1145/3576915.3623169>
54. Yang, Y., Heath, D., Hazay, C., Kolesnikov, V., Venkatasubramanian, M.: Tight zk cpu: Batched zk branching with cost proportional to evaluated instruction. Cryptology ePrint Archive, Paper 2024/456 (2024), <https://eprint.iacr.org/2024/456>
55. Yang, Y., Heath, D., Kolesnikov, V., Devecsery, D.: EZEE: epoch parallel zero knowledge for ANSI C. In: 7th IEEE European Symposium on Security and Privacy, EuroS&P 2022, Genoa, Italy, June 6-10, 2022. pp. 109–123. IEEE, Genoa, Italy (2022). <https://doi.org/10.1109/EuroSP53844.2022.00015>
56. Yao, A.C.C.: How to generate and exchange secrets (extended abstract). In: 27th FOCS. pp. 162–167. IEEE Computer Society Press, Toronto, Ontario, Canada (Oct 27–29, 1986). <https://doi.org/10.1109/SFCS.1986.25>
57. Zippel, R.: Probabilistic algorithms for sparse polynomials. In: International symposium on symbolic and algebraic manipulation. pp. 216–226. Springer (1979)



FLI: Folding Lookup Instances

Albert Garreta^(✉) and Ignacio Manzur

Nethermind Research, London, UK
albert@nethermind.io

Abstract. We introduce two folding schemes for lookup instances: FLI and FLI+SOS. Both use a PIOP to check that a matrix has elementary basis vectors as rows, with FLI+SOS adding a twist based on Lasso’s [26] SOS-decomposability. FLI takes two lookup instances $\{\mathbf{a}_1\}, \{\mathbf{a}_2\} \subseteq \{\mathbf{t}\}$, and expresses them as matrix equations $M_i \cdot \mathbf{t}^\top = \mathbf{a}_i^\top$ for $i = 1, 2$, where each matrix $M_i \in \mathbb{F}^{m \times N}$ has rows which are elementary basis vectors in \mathbb{F}^N . Matrices that satisfy this condition are said to be in \mathbf{R}_{elem} . Then, a folding scheme for \mathbf{R}_{elem} into a relaxed relation is used, which combines the matrices M_1, M_2 as $M_1 + \alpha M_2$ for a random $\alpha \in \mathbb{F}$. Finally, the lookup equations are combined as $(M_1 + \alpha M_2) \cdot \mathbf{t}^\top = (\mathbf{a}_1 + \alpha \mathbf{a}_2)^\top$. In FLI, only the property that a matrix is in \mathbf{R}_{elem} is folded, and this makes the FLI folding step the cheapest among existing solutions. The price to pay is in the cost for proving accumulated instances. FLI+SOS builds upon FLI to enable folding of large SOS-decomposable [26] tables. This is achieved through a variation of Lasso’s approach to SOS-decomposability, which fits FLI naturally. For comparison, we describe (for the first time to our knowledge) straightforward variations of Prostar [5] and Proofs for Deep Thought [7] that also benefit from SOS-decomposability. We see that for many reasonable parameter choices, and especially those arising from lookup-based zkVMs [1], FLI+SOS can concretely be the cheapest folding solution.

Keywords: Folding schemes · Lookup Arguments · SNARKs · zero-knowledge Virtual Machines · Interactive Proofs · Interactive Oracle Proofs

1 Introduction

Folding schemes have been an active area of research in the last few years [4–8, 15, 20, 21]. Informally, these schemes can be described as interactive proofs in which a Prover and a Verifier create a new instance-witness pair for a certain relation \mathbf{R}_2 from two instances-witness pairs for relations $\mathbf{R}_1, \mathbf{R}_2$. The validity of the newly created instance-witness pair implies the validity of the two original instance-witness pairs. The idea is that if this combination process is less expensive than directly proving that the two instance-witness pairs belong to the relevant relations (in Prover time, memory requirements, or proof size), one can save on costs by reducing the task of proving that many instance-witness

pairs belong to \mathbf{R}_1 to proving that a single pair belongs to \mathbf{R}_2 . Initially, these schemes were created with the intention of improving the construction of primitives like Incrementally Verifiable Computation (IVC) [28] and Proof-Carrying-Data (PCD) [10].

Another active area of research is that of lookup arguments: these are arguments that allow a Prover to convince a Verifier that all elements in a vector \mathbf{a} appear in a pre-established vector \mathbf{t} . The vectors \mathbf{t} is often referred to as a *lookup table*, and we use the phrase “look \mathbf{a} into \mathbf{t} ” to mean engaging in the lookup argument to convince a Verifier that all elements in \mathbf{a} appear in \mathbf{t} . Lookup arguments have become very popular within the context of SNARKs [18, 22], because they allow for an efficient treatment of operations that are otherwise difficult to arithmetize. In SNARKs, these operations could be non-native field arithmetic, elliptic curve operations, binary operations, and so on. With lookup arguments, the Prover can use a lookup into a lookup table for the corresponding operation (this table is usually pre-defined), instead of needing to represent the computation in the arithmetization of the SNARK. Recent work in the field of lookup arguments can be essentially split between lookup arguments that use matrix equations [26, 29, 30] and those that use logarithmic derivatives [14, 17, 24]. Lasso [26] is the state-of-the-art matrix-based lookup argument without pre-processing, provided that the lookup table \mathbf{t} has a specific structure, called *SOS-decomposability*. This informally says that to find an entry of the lookup table \mathbf{t} , one can evaluate a multilinear polynomial g at the entries of smaller tables $\mathbf{t}_1, \dots, \mathbf{t}_\alpha$ in some structured way (for details, see Sect. 3.3). The typical example is a range-check table. For instance, to show that a field element is smaller than 2^{256} , one can break the element into 64 (or 32, or 16, and so on) bit parts, and perform a range check for each of these parts. Note that it is not even possible to materialize the table of all elements smaller than 2^{256} . Hence SOS-decomposability gives the ability to perform lookups into gigantic tables.

In a companion paper to Lasso, called Jolt [1], the authors construct a lookup-based zero-knowledge Virtual Machine (zkVM). This realizes an idea sketched out in a ZKResearch blogpost [3] called the *lookup singularity*, which pushes to the extreme the idea of using lookup arguments for verifying computation: instead of only verifying those computations which are expensive or difficult to arithmetize with lookups, all operations are verified with lookups. In [1] the authors show that almost all operations of the RISC-V ISA are SOS-decomposable, and so using Lasso they are able to construct a lookup-based zkVM for the RISC-V instruction set. In a follow-up blogpost and talk [2, 27], the authors express that one of the main steps in Jolt’s roadmap is to implement *continuations*. This means breaking the CPU execution into chunks, and aggregating the proof of correctness of each chunk. The motivation for doing so is to reduce the peak memory consumption of generating a proof, but it comes at the cost of increasing the proof size (since now there is a proof for each chunk). This also leads to the need for proving many polynomial evaluations (several per chunk). As outlined in [27], the authors plan to avoid these problems in two ways (each with its own use cases): one by using recursion and the polynomial com-

mitment scheme from Binius [12,13], and the second by using folding schemes to aggregate the chunks.

1.1 Our Contributions

In this paper, inspired by the use case of continuations in lookup-based zkVMs, we develop two folding schemes for lookup instances (see Definition 2) called FLI and FLI+SOS. The latter is an extension of FLI that is capable of leveraging SOS-decomposability of the lookup table \mathbf{t} . We use two main technical ingredients: a variation of the way Lasso uses SOS decompositions, and a PIOP and a folding scheme for the relation that a matrix has elementary basis vectors as rows (i.e. each row consists entirely of 0, except for one entry being 1).

Say we want to prove that the elements in $\mathbf{a} \in \mathbb{F}^m$ appear in $\mathbf{t} \in \mathbb{F}^N$. We refer to \mathbf{a} as the small table, and to \mathbf{t} as the big table. There are two types of costs that we focus on: the folding costs, and the cost of proving accumulated instances. The former is the Prover and Verifier cost (in field/group operations, random oracle costs, and so on) associated with the folding scheme, and the latter is the Prover and Verifier cost of a SNARK for the accumulated relation.

Irrespective of whether \mathbf{t} is SOS-decomposable (Definition 3) or not, FLI has the cheapest folding Prover and Verifier (among the schemes described in Sect. 1.2), see Table 1 to 3 and Sect. 6. The Prover folding costs of FLI are linear on m , i.e. the size of the small table \mathbf{a} . However, as is the case with the rest of analysed schemes, proving an accumulated instance requires incurring a cost of $O(N)$ at least (recall N is the size of \mathbf{t}). Hence, FLI, as well as the rest of folding schemes for lookup instances, cannot reasonably handle gigantic tables \mathbf{t} .

With this in mind we consider the case when \mathbf{t} is SOS-decomposable, and design a variation of FLI, called FLI+SOS, which leverages SOS-decomposability of \mathbf{t} in a natural way. This enables folding lookup instances where \mathbf{t} is gigantic but SOS-decomposable, without incurring $O(N)$ costs, neither in the folding step, nor when proving accumulated instances. To compare FLI+SOS, we describe straightforward variations of Protostar [5] and Proofs for Deep Thought (abbreviated DT) [7] that make use of the SOS decomposability of \mathbf{t} . See Sect. 1.2 for more details. We emphasize that as is, neither Protostar nor DT support SOS decompositions. To our knowledge, we are the first to describe the variations of these schemes that are compatible with SOS decompositions, and we make a number of favorable assumptions regarding their costs when comparing them to FLI+SOS. We call these variations Protostar+SOS and DT+SOS.

When \mathbf{t} is SOS-decomposable into $\alpha = k \cdot c$ tables of size $N^{1/c}$, we show in Sect. 6 that for choices of m, N, c, k that naturally arise in the context of lookup-based zkVMs, FLI+SOS can overall be the cheapest folding scheme for lookup instances. Therefore it is a candidate for implementing continuations by folding computation chunks. For instance, we show that for $m = 2^{17}, N = 2^{1024}, c \sim 256, \alpha = 2c$ (this particular choice seems to be perfectly plausible in practice when using Jolt [1,27]), with $n_f = 2^3$ foldings then:

- FLI+SOS’s folding Prover is more than $4\times$ cheaper than Protostar+SOS’s, and is orders of magnitude cheaper than DT+SOS’s. In general if $\alpha = k \cdot c$,

then FLI+SOS’s folding Prover is more than $2 \cdot k \times$ cheaper than Protostar+SOS’s.

- FLI+SOS’s folding Verifier is comparable to (but slightly cheaper than) Protostar+SOS’s Verifier, but much cheaper than DT+SOS’s.
- FLI+SOS’s folding Verifier has the lowest random oracle query costs. This is particularly relevant in the context of IVC [28], where the folding Verifier should be represented recursively in a circuit. Each random oracle query could potentially represent numerous and complicated constraints, as some hash functions that heuristically instantiate the random oracle are difficult to arithmetize.

We describe a custom SNARK for the accumulated/relaxed lookup relation in FLI+SOS, namely $\mathbf{R}^{\text{accSOS}}$ (see Sect. 5.4). In our comparison, still with the same parameter values for m, N, c and k , we find that:

- Putting opening proofs for multilinear polynomials to the side, the Prover for accumulated instances of FLI+SOS is around 1.2 times more expensive than that of Protostar+SOS and around $1.3 \times$ more expensive than that of DT+SOS. However, in this regime, DT+SOS’s folding Prover is prohibitively expensive. *We emphasize that this result is obtained while making a number of optimistic simplifications regarding the cost of proving accumulated instances with Protostar+SOS and DT+SOS.* The improvement could be way sharper, cf. Sect. 6 and in particular Remark 3.
- The polynomial opening proof cost of FLI+SOS is around $2 \times$ less than Protostar+SOS and DT+SOS’s optimistic cost for proving accumulated instances. We emphasize that this estimate assumes a naive curve-based (MSM) commitment scheme such as PST [9, 23] (i.e. a “multilinear KZG”), and that one can choose alternative schemes which removes this overhead, at the expense of increasing the folding Verifier work. For example, #2 in [27] proposes using a tensor-like variation of Zeromorph [19] or HyperKZG [25] which would make this step no longer be the bottleneck in FLI+SOS, and would increase the Verifier work by $O(N^{1/c})$ group operations. With this, FLI+SOS’s Verifier would be comparable to DT+SOS’s and more expensive than Protostar+SOS’s. However, FLI+SOS would be the cheapest scheme both in terms of the folding Prover work and the cost of proving accumulated instances.

For more details about the comparison, see Sect. 6 and Table 1 to 4.

Potential Usability in Lattices. Finally, we remark that FLI is based solely on the sumcheck protocol, and because of that, it could potentially be used in the context of lattice-based cryptography. This is in contrast to Protostar and DT, which rely on field-based identities involving logarithmic derivatives [17, 24] which do not seem to carry over to lattices.

1.2 Related Work

To the best of our knowledge, there are three available approaches to folding lookup instances. Hypernova [20] describes one such scheme in which the Prover

Table 1. Comparison of FLI with other folding schemes for lookup instances. The costs are organized in commitment, group exponentiation, and field multiplication costs. We also display the number of rounds of each scheme. This coincides with the number of challenges sent by the Verifier. m and N denote the size of the small table \mathbf{a} on of the big table, respectively. In the “commit” column, a pair (n, S) refers to a commitment of a vector of size n with entries in the set S . FLI has two commitment cost profiles: one is average case, while the other is worst case (cf. Sect. 2 for further details). Here n_f we denote the number of foldings performed so far, u is a small parameter (denoted $c - 1$ in [7]), and M denotes the maximum size of an entry in \mathbf{t} . The cost P_{sps} refers to the field operation cost of running DT’s underlying special sound protocol. In Appendix B this cost is approximated to $\alpha \cdot 19 \max\{m, N^{1/e}\}$ field multiplications. The cost L refers to the cost of computing the coefficients of the polynomial $e(X)$ in [7]. The efficiency of Protostar is displayed for their special-sound version of the logUp lookup argument. The efficiency of Hypernova is displayed for the lookup argument `nlookup` in [20]. When it comes to DT, we consider only the variant built upon logUp-GKR.

Scheme	Prover work			Verifier work		Rounds
	commit	group	field	group	field	
Protostar [5]	$(2m, \mathbb{F}), (m, [m])$	7	$O(m)$	3	$O(1)$	1
Hypernova [20]	–	–	$O(N)$	–	$O(m \log N)$	$\log(m) + O(1)$
Deep Thought [7]	$(3m, [M])$	$u \log N$	$\begin{cases} O(m \log(m)) + \\ + P_{\text{sps}} + L \end{cases}$	$u \log N$	$u \log(N)$	$u \log(N)$
FLI (this work)	$\begin{cases} \text{avg: } (\rho, \mathbb{F}), (m - \rho, \mathbb{B}) \\ \text{worse: } (m, \mathbb{F}) \\ \rho := \min\{mn_f/N, m\} \end{cases}$	4	$O(m)$	4	$O(1)$	1

cost is $O(N)$, while the Verifier’s work is $O(m \log(N))$, where here N is the size of the big table \mathbf{t} , and m is the size of the small table \mathbf{a} . Protostar [5] describes a folding scheme based on the logUp lookup argument [17] with Prover’s costs $O(m)$, with the concrete costs being rather large due to the need of committing to 2 size- m vectors with entries of arbitrary length. A related posterior work, Proofs for Deep Thought (DT in short) [7], presents an alternative folding scheme for lookup instances in which the Prover’s costs depend only on m . On the other hand, DT’s Verifier has a larger cost than Protostar’s, cf. Table 1. Since we are interested in folding schemes with a Prover sublinear on N and a succinct Verifier, we compare FLI mostly with Protostar and DT.

As we mentioned, FLI+SOS can naturally leverage the SOS decomposability of the big table \mathbf{t} , and as far as we are aware, FLI+SOS is the first of its kind in this sense. It is easy, however, to envision ways in which the previously mentioned schemes (Hypernova, Protostar, and DT) can also exploit SOS decomposability. Namely, one can first run the first step of Lasso (see Sect. 2), which splits a lookup instance into α smaller lookup instances, and then use either scheme to fold the α instances into α accumulated instances. We refer to this variation as $\{\text{Scheme}\} + \text{SOS}$, where “Scheme” can be any folding scheme for lookup instances, e.g. Protostar, or DT. We remark that proving accumulated instances for these schemes (Protostar + SOS and DT+SOS) is not a straightforward task

Table 2. Comparison of the Prover in FLI+SOS with the Prover of other folding schemes when the big table \mathbf{t} is SOS-decomposable (Definition 3). “Protostar + SOS” and “Deep Thought (DT) + SOS” both refer to first performing Lasso’s SOS reduction and then applying Protostar or [7], respectively, to the resulting α lookup instances of a table of size m into a table of size $N^{1/c}$. We follow the same notation as in Table 1. $N^{1/c}$ is the size of the SOS decomposed tables (cf. Sect. 3.3); $\alpha = k \cdot c$, where k is a small constant (typically 1 or 2); and g is the multilinear polynomial providing the SOS decomposition and $|g|$ its arithmetic complexity.

Scheme	Prover work		
	commit	group	field
Protostar+SOS	$\alpha \cdot (2m, \mathbb{F}), \alpha \cdot (m, [m])$	7α	$m \deg(g)(\alpha + g)$
Deep Thought+SOS	$\alpha \cdot (3m, [M])$	$\alpha \cdot u \log(N^{1/c})$	$\left\{ \begin{array}{l} O(\alpha m \log(m)) \\ +m \deg(g)(\alpha + g) \\ +\alpha \cdot \mathbf{P}_{\text{sps}} + \alpha \cdot \mathbf{L} \end{array} \right.$
FLI+SOS	$\left\{ \begin{array}{l} \text{avg: } c \cdot (\rho, \mathbb{F}), c \cdot (m - \rho, \mathbb{B}) \\ \text{worse: } c \cdot (m, \mathbb{F}) \\ c \cdot (m, \mathbb{B}) \\ \rho := \min\{mn_f/N^{1/c}, m\} \end{array} \right.$	$4c + 1$	$m \deg(g)(\alpha + g)$

however. For the sake of comparison, we sketch a simplified method in Sect. 1.2 and 6. We compare our folding scheme FLI with Hypernova, Protostar, and DT; and we compare FLI+SOS with Protostar + SOS, and DT + SOS, cf. Table 1 to 4 and Sect. 6.

1.3 Organization of the Paper

Sect. 2 outlines the techniques used to develop FLI. In Sect. 3, we introduce lookup relations and SOS-decomposable tables. Additional definitions on folding schemes, IPs/(P)IOPs, soundness, and the sumcheck protocol as well as effered proofs are in the extended version of the paper. Section 4 constructs a PIOP and folding scheme for relation \mathbf{R}_{elem} (stating that a matrix has elementary basis vectors as rows), which is extended in Sect. 5 to build FLI, incorporating a variation on SOS decomposition. Section 6 shows FLI’s efficiency for key parameters. Further comparisons are in Appendix A, and Prover cost computations in Appendix B.

2 Techniques

Let \mathbf{R} and \mathbf{R}_{acc} be two instance-witness relations. A folding scheme from $\mathbf{R} \times \mathbf{R}_{\text{acc}}$ to \mathbf{R}_{acc} is an interactive protocol between a Prover and a Verifier. The Verifier takes as input a pair of instances $(\mathbf{x}, \mathbf{x}_{\text{acc}}) \in \mathbf{R} \times \mathbf{R}_{\text{acc}}$, and outputs a new instance $\mathbf{x}'_{\text{acc}} \in \mathbf{R}_{\text{acc}}$ at the end of the protocol. One requires that if the Prover knows a valid witness \mathbf{w}'_{acc} for \mathbf{x}'_{acc} , then it knows (except with negligible probability)

Table 3. Comparison of the Verifier in FLI+SOS with the Verifier of other folding schemes when the big table \mathbf{t} is SOS-decomposable (Definition 3). We follow the same terminology and notation as in Table 1 and 2

Scheme	Verifier work		Rounds
	group	field	
Protostar+SOS	3α	$O(\alpha \log(m))$	$\log(m) + \alpha$
Deep Thought+SOS	$\alpha u \log(N^{1/c})$	$O(\alpha \cdot u \log(N^{1/c}) + \log(m))$	$\log(m) + \alpha \cdot u \log(N^{1/c})$
FLI+SOS	$4c + 1$	$O(\alpha \log(m))$	$\log(m) + \alpha$

valid witnesses \mathbf{w} and \mathbf{w}_{acc} for \mathbf{x} and \mathbf{x}_{acc} , respectively. Often, one speaks of folding schemes for \mathbf{R} , omitting \mathbf{R}_{acc} .

Here we consider the *lookup relation* \mathbf{R}_{Look} . For a fixed field \mathbb{F} , an instance-witness pair $(\mathbf{x}; \mathbf{w}) \in \mathbf{R}_{\text{Look}}$ has the form $\mathbf{x} = (m, N, \mathbf{t}, \text{cm}_{\mathbf{t}}, \text{cm}_{\mathbf{a}})$, and $\mathbf{w} = (\mathbf{a})$, where $m, N \in \mathbb{N}$, $\mathbf{t} \in \mathbb{F}^N$, $\mathbf{a} \in \mathbb{F}^m$, and $\text{cm}_{\mathbf{t}}, \text{cm}_{\mathbf{a}}$ are vector commitments to the vectors \mathbf{t}, \mathbf{a} . These vectors are often referred to as *tables*. The lookup instance is *valid* if $\{\mathbf{a}_i \mid i \in [m]\} \subseteq \{\mathbf{t}_i \mid i \in [N]\}$, and $\text{Commit}(\mathbf{t}) = \text{cm}_{\mathbf{t}}$, $\text{Commit}(\mathbf{a}) = \text{cm}_{\mathbf{a}}$ for a fixed commitment scheme Commit (for simplicity we omit referring to the randomness used in the commitments). In other words:

$$\mathbf{R}_{\text{Look}} := \left\{ (\mathbf{x}; \mathbf{w}) = (m, N, \mathbf{t}, \text{cm}_{\mathbf{t}}, \text{cm}_{\mathbf{a}}; \mathbf{a}) \mid \begin{array}{l} \{\mathbf{a}_i \mid i \in [m]\} \subseteq \{\mathbf{t}_i \mid i \in [N]\} \\ \text{Commit}(\mathbf{t}) = \text{cm}_{\mathbf{t}}, \text{Commit}(\mathbf{a}) = \text{cm}_{\mathbf{a}} \end{array} \right\}$$

Informally, in this overview we sometimes denote elements in \mathbf{R}_{Look} by $(\mathbf{t}, \text{cm}_{\mathbf{t}}, \text{cm}_{\mathbf{a}}; \mathbf{a}) \in \mathbf{R}_{\text{Look}}$, omitting any reference to the table sizes m and N . Typically, we assume m and N to be powers of two, with N being much larger than m . Because of this, we often informally call \mathbf{t} the *big table*, and \mathbf{a} the *small table*.

We next describe how FLI works at a high level. We first recall the now standard observation [29, 30] that a lookup instance $(\mathbf{t}, \text{cm}_{\mathbf{t}}, \text{cm}_{\mathbf{a}}; \mathbf{a}) \in \mathbf{R}_{\text{Look}}$ is valid if and only if there exists a $m \times N$ matrix $M \in \mathbb{F}^{m \times N}$ such that:

- $M \cdot \mathbf{t}^T = \mathbf{a}^T$, where T denotes transposition and $M \cdot \mathbf{t}^T$ denotes matrix-vector multiplication.
- Each row of M is an *elementary basis vector*, i.e. it consists only of zeros, except for one entry, which is 1. We define a relation capturing this property:

$$\mathbf{R}_{\text{elem}} := \left\{ (\mathbf{x}; \mathbf{w}) = (m, N, \text{cm}_M; M) \mid \begin{array}{l} \text{All rows of } M \text{ are elem. basis vectors,} \\ \text{Commit}(M) = \text{cm}_M \end{array} \right\}$$

- $\text{Commit}(\mathbf{t}) = \text{cm}_{\mathbf{t}}$ and $\text{Commit}(\mathbf{a}) = \text{cm}_{\mathbf{a}}$.

Table 4. Dominant costs of the protocols for proving accumulated instances with FLI and FLI+SOS (cf. 5.4). We follow the same notation as in Tables 1 to 3. Besides that, $s \leq mN^{1/c}$ denotes the sparsity of the accumulated matrices $M_{\text{acc}}, E_{\text{acc}}, M_i^{\text{acc}}$. The second and third column shows the dominant Prover and Verifier cost in field multiplications. In the Openings column, the notation $v \cdot (a\text{-variate}, b\text{-sparse})$ refers to v opening proofs of a -variate multilinear polynomials that are b -sparse (i.e. at most b of their evaluations in $\{0, 1\}^{\log(a)}$ are nonzero). The Verifier opening proof costs are not reflected in the table. By ‘‘SOS’’ we mean that the multilinear polynomial is a small table resulting from a SOS decomposition. These can often be evaluated in $\log(N^{1/c})$ time, and hence the opening can be computed directly by the Verifier, rather than proved. By ‘‘dense’’ we mean that the polynomial can potentially take nonzero values on all the hypercube.

Scheme	Prover field work	Verifier field work	Openings
FLI	$(2m + 1)N + s(\log(m) + 3)$	$O(\log(mN))$	$1 \cdot (\log(mN)\text{-var}, \nu\text{-sparse})$ $1 \cdot (\log(N)\text{-var}, \text{SOS})$ $1 \cdot (\log(m)\text{-var}, \text{dense})$ $\nu := \min\{n_fm, mN\}$
FLI+SOS	$(2m + 5\alpha + 1)N^{1/c} + (\alpha + 2)s + 2m$	$O(\log(mN^{1/c}))$	$1 \cdot (\log(mN^{1/c})\text{-var}, \nu\text{-sparse})$ $1 \cdot (\log(N^{1/c})\text{-var}, \text{SOS})$ $1 \cdot (\log(m)\text{-var}, \text{dense})$ $\nu := \min\{n_fm, mN^{1/c}\}$

One can then define:

$$\mathbf{R}_{\text{MLook}} := \left\{ \left(\begin{array}{c} \mathbf{x}; \\ \mathbf{w} \end{array} \right) = \left(\begin{array}{c} m, N, \mathbf{t}, \text{cm}_{\mathbf{t}}, \text{cm}_{\mathbf{a}}, \text{cm}_M; \\ \mathbf{a}, M \end{array} \right) \left| \begin{array}{l} M \in \mathbb{F}^{m \times N} \\ (m, N, \text{cm}_M; M) \in \mathbf{R}_{\text{elem}} \\ M \cdot \mathbf{t}^\top = \mathbf{a}^\top \\ \text{Commit}(\mathbf{t}) = \text{cm}_{\mathbf{t}}, \\ \text{Commit}(\mathbf{a}) = \text{cm}_{\mathbf{a}} \end{array} \right. \right\}$$

As with \mathbf{R}_{Look} , we will omit referring to m, N when talking about instance-witness pairs from $\mathbf{R}_{\text{MLook}}$, and we proceed similarly with \mathbf{R}_{elem} . Let $(\mathbf{x}_i; \mathbf{w}_i) = (\mathbf{t}, \text{cm}_{\mathbf{t}}, \text{cm}_{\mathbf{a}_i}, \text{cm}_{M_i}; \mathbf{a}_i, M_i)$ ($i = 1, 2$) be two instance-witness pairs from $\mathbf{R}_{\text{MLook}}$ that we wish to fold. By definition, we have $M_1 \cdot \mathbf{t}^\top = \mathbf{a}_1^\top$ and $M_2 \cdot \mathbf{t}^\top = \mathbf{a}_2^\top$. We visualize such instances as:

$$\left\{ \begin{array}{l} M_1 \cdot \mathbf{t}^\top = \mathbf{a}_1^\top \\ (\text{cm}_{M_1}; M_1) \in \mathbf{R}_{\text{elem}} \end{array} \right\} \quad \left\{ \begin{array}{l} M_2 \cdot \mathbf{t}^\top = \mathbf{a}_2^\top \\ (\text{cm}_{M_2}; M_2) \in \mathbf{R}_{\text{elem}} \end{array} \right\} \quad (1)$$

Note that, fixing \mathbf{t} , the first part of the instances in (1) are linear constraints on the matrices M_1, \mathbf{a}_1 and M_2, \mathbf{a}_2 . Hence, a natural step towards folding the instances (1) is to have the verifier send a random challenge $\alpha \leftarrow \mathbb{F}$, and then merge (1) into a single claim of the form

$$\begin{cases} (M_1 + \alpha M_2) \cdot \mathbf{t}^\top = \mathbf{a}_1^\top + \alpha \mathbf{a}_2^\top \\ (\text{cm}_{M_1}; M_1), (\text{cm}_{M_2}; M_2) \in \mathbf{R}_{\text{elem}} \end{cases} \quad (2)$$

The next natural step is to apply a folding scheme for the relation \mathbf{R}_{elem} , so that the two claims above, namely $(\text{cm}_{M_1}; M_1) \in \mathbf{R}_{\text{elem}}$ and $(\text{cm}_{M_2}; M_2) \in \mathbf{R}_{\text{elem}}$, can be folded into an instance of an accumulated version of the relation \mathbf{R}_{elem} . One way to do this is by first noting that $(\text{cm}_M; M)$ belongs to \mathbf{R}_{elem} if and only if:

- $M_{ij}^2 = M_{ij}$ for all matrix entries M_{ij} of M ($i \in [m], j \in [N]$). This property is equivalent to saying that all entries of M are either 0 or 1.
- $M \cdot \mathbf{1}^\top = \mathbf{1}^\top$, where $\mathbf{1}$ is the N -dimensional vector consisting entirely of 1's. Together with the above property, this ensures that all rows of M are elementary vectors¹.
- $\text{Commit}(M) = \text{cm}_M$.

By looking at M as a mN -dimensional witness vector, we reformulate the above conditions as a R1CS-type constraint. Then, we use a Nova-like approach [21] so as to obtain a folding scheme for the relation \mathbf{R}_{elem} into a relaxed version of \mathbf{R}_{elem} , which we denote $\mathbf{R}_{\text{elem}}^{\text{acc}}$. Namely $\mathbf{R}_{\text{elem}}^{\text{acc}}$ incorporates a mN -dimensional error vector E , a slackness parameter μ , and a commitment cm_E to E . Then $(\mu, \text{cm}_M, \text{cm}_E; M, E) \in \mathbf{R}_{\text{elem}}^{\text{acc}}$ if and only if $M_{ij}^2 = M_{ij} + E_{ij}$ for all $i \in [m], j \in [N]$, $M \cdot \mathbf{1}^\top = (1 + \mu) \cdot \mathbf{1}^\top$, and the commitments cm_M and cm_E are commitments to M and E . Formally,

$$\mathbf{R}_{\text{elem}}^{\text{acc}} := \left\{ \left(\begin{array}{c} \mathbf{x}; \\ \mathbf{w} \end{array} \right) = \left(\begin{array}{c} m, N, \mu, \text{cm}_M, \text{cm}_E; \\ M, E \end{array} \right) \left| \begin{array}{l} M \circ M = M + E \\ M \cdot \mathbf{1}^\top = (1 + \mu) \cdot \mathbf{1}^\top, \\ \text{Commit}(M) = \text{cm}_M, \text{Commit}(E) = \text{cm}_E \end{array} \right. \right\}$$

Here \circ denotes the Hadamard (i.e. component-wise) product. Given $(\text{cm}_M; M) \in \mathbf{R}_{\text{elem}}$ and $(\mu, \text{cm}_{M_{\text{acc}}}, \text{cm}_E; M_{\text{acc}}, E) \in \mathbf{R}_{\text{elem}}^{\text{acc}}$:

1. The Prover computes a commitment cm_T to an intermediate cross term $T = 2(M_{\text{acc}} \circ M) - M$ and sends cm_T to the Verifier;
2. The Verifier replies with a random challenge $\alpha \leftarrow \mathbb{F}$;
3. Both Prover and Verifier output $\text{cm}_{M'_{\text{acc}}} = \text{cm}_{M_{\text{acc}}} + \alpha \text{cm}_M$, $\text{cm}_{E'} = \text{cm}_E + \alpha \text{cm}_T + \alpha^2 \text{cm}_M$, $\mu' = \mu + \alpha$;
4. The Prover additionally outputs $M'_{\text{acc}} = M_{\text{acc}} + \alpha M$, $E' = E + \alpha T + \alpha^2 M$.

As a result, FLI's folding Verifier is very simple, only performing a handful of operations with a given vector commitment. When using, say, curve-based commitment schemes, this translates to 4 group additions and 4 scalar multiplications. Note that FLI only needs one random oracle query. The folding proof size consist in one group and field elements (we don't count $\text{cm}_{M'_{\text{acc}}}, \text{cm}_{E'}$ as part of the proof).

When it comes to Prover costs, note that the matrix $M \in \mathbb{F}^{m \times N}$ (which we look at as a vector of size mN) is m -sparse, meaning that all its entries

¹ Here one needs to assume that the characteristic of \mathbb{F} is larger than N .

except m are 0. In fact, all nonzero entries of M are 1. Consequently, the matrix $M_{\text{acc}} \circ M$ is also m -sparse, and so is the vector T . As such, standard curve-based commitment schemes allow to commit to M and to T in time $O(m)$. For example, the PST² [9, 23] scheme can commit to M with exactly $m - 1$ group additions.

On the other hand, in general, nonzero elements in T can have arbitrary size. This is because M_{acc} has entries that are computed from the Verifier’s previous folding challenges, which were sampled randomly in \mathbb{F} . The concrete cost of committing to T in the *worst case* can be relatively large, e.g. $\approx 2^8 m$ or $2^9 m$ when using PST and Pippenger’s algorithm to compute Multi Scalar Multiplications (MSM) (see, for example, the benchmarks in Table 1 of [17]). However, we remark that the above is a *worst case*. Since $M \in \mathbf{R}_{\text{elem}}$, if M_{acc} is very sparse (as is when not many folding steps have been performed), then $M_{\text{acc}} \circ M$ is even more sparse with high likelihood. If we think of M as being randomly sampled in \mathbf{R}_{elem} , then the number of nonzero elements in $M_{\text{acc}} \circ M$ is, in expectation, $\leq mn_f/N$, where n_f is the number of folding steps performed so far. Thus, when $n_f m \ll N$, we have that $M_{\text{acc}} \circ M$ is essentially the zero vector, on average. In that case, **FLI’s Prover only commits to sparse vectors containing almost exclusively small entries.**

Lasso’s SOS decomposability. We recall SOS-decomposability, one of the key ideas of the Lasso and Jolt papers [1, 26]. Formally, a table $\mathbf{t} \in \mathbb{F}^N$ is *SOS-decomposable* if there exists $\alpha := k \cdot c$ tables $\mathbf{t}_1, \dots, \mathbf{t}_\alpha$ of size $N^{1/c}$, i.e. $\mathbf{t}_i \in \mathbb{F}^{N^{1/c}}$ for all i , and an α -variate multilinear polynomial g such that:

$$\forall \mathbf{y} \in \mathbb{B}^{\log N}, t(\mathbf{y}) = g \left(\begin{array}{c} \mathbf{t}_1(\mathbf{y}_1), \dots, \mathbf{t}_k(\mathbf{y}_1), \mathbf{t}_{k+1}(\mathbf{y}_2), \dots, \mathbf{t}_{2k}(\mathbf{y}_2), \dots \\ \dots, \mathbf{t}_{k \cdot c - 1}(\mathbf{y}_c), \mathbf{t}_\alpha(\mathbf{y}_c) \end{array} \right) \quad (3)$$

Here, we let $\mathbb{B} = \{0, 1\}$, and we index the entries of \mathbf{t} with elements from the hypercube $\mathbb{B}^{\log N}$, denoting $\mathbf{t}(\mathbf{y})$ the entry of \mathbf{t} indexed by the element $\mathbf{y} \in \mathbb{B}^{\log(N)}$. Further, $\mathbf{y}_1, \dots, \mathbf{y}_c$ are all vectors from $\mathbb{B}^{\log(N)/c}$ such that $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_c)$. For the applications mentioned in Lasso and Jolt [1, 26], α is c or a small multiple of c . As exemplified by the Jolt paper [1], many natural tables \mathbf{t} are SOS-decomposable (e.g., tables containing RISC-V instructions). Importantly for this paper, c can be chosen as large as wanted.

FLI and its natural use of SOS decomposability (FLI+SOS). One of the contributions of this work is a variation of SOS decompositions that blends seamlessly with our folding approach. We emphasize that this does not simply consist in performing Lasso’s SOS decomposition and then folding the resulting smaller lookup instances. The starting observation is that when \mathbf{t} is SOS-decomposable, the statement that $(\mathbf{t}, \mathbf{cm}_{\mathbf{t}}, \mathbf{cm}_{\mathbf{a}}; \mathbf{a}) \in \mathbf{R}_{\text{Look}}$ is equivalent (leaving aside the constraints involving $\mathbf{cm}_{\mathbf{t}}$ and $\mathbf{cm}_{\mathbf{a}}$) to the existence of $m \times N^{1/c}$ matrices M_1, \dots, M_c with elementary vectors as rows such that:

² Here we refer to the PST version from [9] which uses the Lagrange basis instead of the monomial basis.

$$\forall \mathbf{x} \in \mathbb{B}^{\log(m)}, \mathbf{a}(\mathbf{x}) = g \left(\sum_{\mathbf{y}} M_1(\mathbf{x}, \mathbf{y}) \cdot \mathbf{t}_1(\mathbf{y}), \dots, \sum_{\mathbf{y}} M_c(\mathbf{x}, \mathbf{y}) \cdot \mathbf{t}_c(\mathbf{y}) \right) \quad (4)$$

where \mathbf{y} runs over $\mathbb{B}^{\log(N^{1/c})}$. Indeed, one has that $(\mathbf{t}, \mathbf{cm}_{\mathbf{t}}, \mathbf{cm}_{\mathbf{a}}; \mathbf{a}) \in \mathbf{R}_{\text{Look}}$ if and only if for all $\mathbf{x} \in \mathbb{B}^{\log(m)}$, there exists a $\mathbf{y} \in \mathbb{B}^{\log(N)}$ such that $\mathbf{a}(\mathbf{x}) = \mathbf{t}(\mathbf{y})$. But Eq. (3) implies that $\mathbf{t}(\mathbf{y}) = g(\mathbf{t}_1(\mathbf{y}_1), \dots, \mathbf{t}_k(\mathbf{y}_1), \mathbf{t}_{k+1}(\mathbf{y}_2), \dots, \mathbf{t}_c(\mathbf{y}_c))$, where $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_c) \in \mathbb{B}^{\log(N)/c}$. The matrices M_1, \dots, M_c respectively indicate, for each $\mathbf{x} \in \mathbb{B}^{\log(m)}$, the indices $\mathbf{y}_1, \dots, \mathbf{y}_c$ such that $\mathbf{a}(\mathbf{x}) = g(\mathbf{t}_1(\mathbf{y}_1), \dots, \mathbf{t}_k(\mathbf{y}_1), \mathbf{t}_{k+1}(\mathbf{y}_2), \dots, \mathbf{t}_c(\mathbf{y}_c))$ holds, i.e. for each $\mathbf{x} \in \mathbb{B}^{\log(m)}$ and $i \in [c]$, the row of M_i indexed by the element \mathbf{x} consists of zeros everywhere except for a one in the column indexed by \mathbf{y}_i . This way, $\sum_{\mathbf{y}} M_i(\mathbf{x}, \mathbf{y}) \cdot \mathbf{t}_j(\mathbf{y}) = \mathbf{t}_j(\mathbf{y}_i)$, for all $j \in [k]$. Above and below, we look at $\mathbf{a}, M_i, \mathbf{t}_i$ as the multilinear extensions (MLE) (cf. Sect. 3.1) of the corresponding vectors.

With (4) in mind, we describe an extension of FLI that can handle SOS decomposability. Say we wish to fold two instances $(\mathbf{t}, \mathbf{cm}_{\mathbf{t}}, \mathbf{cm}_{\mathbf{a}_i}; \mathbf{a}_i) \in \mathbf{R}_{\text{Look}}$, $i = 1, 2$. The protocol, which we call FLI+SOS, proceeds as follows:

- P starts off by committing to the m -sparse matrices $M_{1,1}, \dots, M_{1,c}$ and $M_{2,1}, \dots, M_{2,c}$ such that (4) holds, respectively, for the instances $(\mathbf{t}, \mathbf{cm}_{\mathbf{t}}, \mathbf{cm}_{\mathbf{a}_1}; \mathbf{a}_1)$ and $(\mathbf{t}, \mathbf{cm}_{\mathbf{t}}, \mathbf{cm}_{\mathbf{a}_2}; \mathbf{a}_2)$. Let $\mathbf{cm}_{M_{i,j}}$ be the commitments. Then $(\mathbf{cm}_{M_{i,j}}; M_{i,j}) \in \mathbf{R}_{\text{elem}}$.
- Next for both $i = 1, 2$, P and V run a sumcheck protocol to assert the equality:

$$\sum_{\mathbf{x} \in \mathbb{B}^{\log(m)}} \left(\mathbf{a}_i(\mathbf{x}) - g \left(\sum_{\mathbf{y}} M_{i,1}(\mathbf{x}, \mathbf{y}) \cdot \mathbf{t}_1(\mathbf{y}), \dots, \sum_{\mathbf{y}} M_{i,c}(\mathbf{x}, \mathbf{y}) \cdot \mathbf{t}_c(\mathbf{y}) \right) \right) \tilde{\mathbf{e}}\mathbf{q}(\boldsymbol{\beta}, \mathbf{x}) = 0$$

where $\boldsymbol{\beta} \in \mathbb{F}^{\log(m)}$ is a random challenge from the Verifier. This equality ensures that, except with negligible probability, (4) holds for $i = 1, 2$.

- At the end of the sumcheck protocol, P and V are left with evaluation claims of the following form, for $j \in [c]$, $i \in \{1, 2\}$, and $(j - 1)k + 1 \leq \ell \leq jk$:

$$\mathbf{a}_i(\mathbf{r}) = d, \quad \sum_{\mathbf{y}} M_{i,j}(\mathbf{r}, \mathbf{y}) \cdot \mathbf{t}_\ell(\mathbf{y}) = c_{ij\ell},$$

- The Verifier sends a random challenge $\alpha \in \mathbb{F}$, and the resulting folded instance is

$$\begin{aligned} (\mathbf{a}_1 + \alpha \mathbf{a}_2)(\mathbf{r}) &= d', \quad (\mathbf{cm}_{M_{i,j}}; M_{i,j}) \in \mathbf{R}_{\text{elem}} \\ \sum_{\mathbf{y}} (M_{1,j} + \alpha M_{2,j})(\mathbf{r}, \mathbf{y}) \cdot \mathbf{t}_\ell(\mathbf{y}) &= c'_{j\ell}, \quad j \in [c], (j - 1)k + 1 \leq \ell \leq jk \end{aligned}$$

for random $\mathbf{r} \in \mathbb{F}^{\log(m)}$ and some field elements $d', c'_{j\ell}$.

- As we explained, the claims $(\mathbf{cm}_{M_{i,j}}; M_{i,j}) \in \mathbf{R}_{\text{elem}}$ can be expressed as R1CS instances, and folded in a Nova-like fashion.

This is a simplified description of FLI+SOS where both instances are lookup instances. In practice, we fold lookup instances into a relaxed lookup relation that we describe in Sect. 5. Further, we use a single matrix to accumulate all the claims of the form $(\mathbf{cm}_{M_{i,j}}; M_{i,j}) \in \mathbf{R}_{\text{elem}}$, and in parallel we fold the evaluation claims. The folding Prover and Verifier of FLI+SOS are still the cheapest (see Tables 2 and 3). We also show by using an example that the cost of proving accumulated instances with FLI is the cheapest in relevant scenarios (Sect. 6).

Proving Accumulated Instances. We describe custom PIOPs that allow to prove that an instance accumulated with FLI+SOS (or FLI) is valid. These protocols are simple and only consist in sumchecks, cf. Table 4, and Sect. 6. The dominant Prover cost for proving accumulated instances with FLI+SOS is $m(2N^{1/c} + 1) + 3\alpha \cdot m \cdot n_f$ field operations, with the $mN^{1/c}$ cost being due to the fact that we treat the $m \times N^{1/c}$ matrices M_i as dense matrices, and the other cost is related to the sparsity of the accumulated matrices after n_f foldings. The fact that we can freely choose the parameter c makes this Prover cost very similar to the other available options in some practical scenarios, which we discuss in Sect. 6). This is considering very optimistic costs for the protocols that prove accumulated instances with Protostar+SOS or DT+SOS (see Sect. 6, and Remark 3). The final step in the protocols that prove accumulated instances is the opening of certain multivariate polynomials in $\log(m)$ or $\log(mN^{1/c})$ variables at random vectors of field elements. While these openings can be expensive, many of them are at the same vector of field elements and can be batched. By choosing a polynomial commitment scheme carefully (like certain variations [27] of Zeromorph [19]), it is possible to reduce the cost of these openings at the expense of a worse Verifier cost. We discuss this in Sect. 6.

3 Preliminaries

Throughout the document we fix a finite field \mathbb{F} . Given an integer $k \geq 1$ we let $[k] := \{1, \dots, k\}$. We let $\mathbb{B}^k = \{0, 1\}^k := \{(b_1, \dots, b_k) \mid b_i \in \mathbb{B}, \text{ for all } i \in [k]\}$ be the hypercube of dimension k , or, in other words, the set of all sequences of k bits. We next provide formal descriptions pertaining multilinear polynomials, lookup relations, folding schemes, and SOS decomposability. We refer to the full version of this paper for extended preliminaries on interactive proofs, and the sumcheck protocol.

Let $\mathbf{X} = (X_1, \dots, X_n)$ be a vector of variables. We let $\mathbb{F}[\mathbf{X}]$ denote the ring of multivariate polynomials on variables \mathbf{X} and with coefficients in \mathbb{F} . By $\mathbb{F}^{\leq d}[\mathbf{X}]$ we denote the set of polynomials from $\mathbb{F}[\mathbf{X}]$ whose variables have individual degree at most d . For example, $\mathbb{F}^{\leq 1}[\mathbf{X}]$ is the set of multilinear polynomials on variables \mathbf{X} .

We use λ to denote the security parameter. A function $f(\lambda)$ is in $\text{poly}(\lambda)$ if there exists a $c \in \mathbb{N}$ such that $f(\lambda) = O(\lambda^c)$. If for all $c \in \mathbb{N}$, $f(\lambda) = o(\lambda^{-c})$, then $f(\lambda)$ is in $\text{negl}(\lambda)$ and is said to be negligible.

3.1 Multilinear Polynomials

Let $n \geq 1$ and let $\mathbf{X} = (X_1, \dots, X_n)$ be a tuple of variables. It is well-known that a multilinear polynomial $f(\mathbf{X}) \in \mathbb{F}^{\leq 1}[\mathbf{X}]$ is uniquely defined by the multiset of the values it takes on \mathbb{B}^n , i.e. $f(\mathbb{B}^n) := \{f(\mathbf{x}) \mid \mathbf{x} \in \mathbb{B}^n\}$. In other words, any two $f, g \in \mathbb{F}^{\leq 1}[\mathbf{X}]$ such that $f(\mathbf{x}) = g(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{B}^n$ are the same polynomial. Further, given a map $f : \mathbb{B}^n \rightarrow \mathbb{F}$, there always exist a unique multilinear polynomial on n variables, denoted $\tilde{f}(\mathbf{X})$, such that $\tilde{f}(\mathbf{x}) = f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{B}^n$. It is given by the expression

$$\tilde{f}(\mathbf{X}) := \sum_{\mathbf{x} \in \mathbb{B}^n} f(\mathbf{x}) \cdot \tilde{\text{eq}}(\mathbf{x}; \mathbf{X}) \tag{5}$$

where $\tilde{\text{eq}}(\mathbf{x}; X)$ is the unique multilinear polynomial on n variables that takes the value 0 on all points of the hypercube \mathbb{B}^n , except at \mathbf{x} where it takes the value 1. Precisely,

$$\tilde{\text{eq}}(\mathbf{x}; \mathbf{X}) := \prod_{i \in [n]} (x_i X_i - (1 - x_i)(1 - X_i)).$$

This unique multilinear polynomial $\tilde{f}(\mathbf{X})$ is called the *multilinear extension (MLE)* of f . Given a vector $\mathbf{v} = (v_1, \dots, v_N) \in \mathbb{F}^N$, we define the MLE of \mathbf{v} (denoted by $\tilde{\mathbf{v}}(\mathbf{X})$) as the MLE of the map $\mathbf{v} : \mathbb{B}^n \rightarrow \mathbb{F}$ assigning to each element $\mathbf{x} \in \mathbb{B}^n$ the element $v_{\mathbf{x}}$, where here we interpret \mathbf{x} as the natural number whose binary representation is \mathbf{x} .

Throughout the paper we fix a Polynomial Commitment Scheme (PCS) for multilinear polynomials (Setup, Commit, Open, Eval) (cf. the extended version of this paper for the definition of such a PCS).

3.2 Lookup Relations

An *indexed relation* is a subset $\mathbf{R} \subseteq \{0, 1\}^* \times \{0, 1\}^* \times \{0, 1\}^*$. Given $(\mathbf{i}, \mathbf{x}; \mathbf{w}) \in \mathbf{R}$, the string \mathbf{i} is called an *index*, \mathbf{x} is called an *instance*, and \mathbf{w} a *witness*. In this paper we often interpret instances and witnesses as vectors of field elements and indices consisting of tuples of natural numbers and field descriptions, but this need not always be the case. When describing Polynomial IOPs for example, the index and the instance can contain oracles to polynomials, and the witness can contain polynomials.

Throughout the paper we let N denote a “large table” size and $m \leq N$ denote a “small table” size. For simplicity, we assume both N and m are powers of 2. A vector $\mathbf{v} \in \mathbb{F}^k$ is said to be *r-sparse* if \mathbf{v} has at most r entries different than 0.

Definition 1 (Lookup relation and big/small tables). *The lookup relation \mathbf{R}_{Look} is defined as:*

$$\mathbf{R}_{\text{Look}} := \left\{ \left(\begin{array}{l} \mathbf{i} = (\mathbb{F}, N, m), \\ \mathbf{x} = (\mathbf{a}, \mathbf{t}); \\ \mathbf{w} = \emptyset \end{array} \right) \mid \begin{array}{l} N, m \geq 1, \\ \{a(\mathbf{x}) \mid \mathbf{x} \in \mathbb{B}^{\log m}\} \subseteq \{t(\mathbf{y}) \mid \mathbf{y} \in \mathbb{B}^{\log(N)}\}. \end{array} \right\}$$

We call \mathbf{a} a small table, and \mathbf{t} a lookup table (or big table).

Unless stated otherwise, we make no assumption on the number of repeated values in \mathbf{t} . I.e. \mathbf{t} may have repeated values.

Definition 2 (Lookup instances). We call a tuple of the form $((\mathbb{F}, N, m), (\mathbf{a}, \mathbf{t}))$ with $\mathbf{t} \in \mathbb{F}^N$, $\mathbf{a} \in \mathbb{F}^m$ a lookup instance. Sometimes the index (\mathbb{F}, N, m) is omitted. An instance may or may not belong to $\mathbf{R}_{\text{Lookup}}$.

Committed Matrix Lookup Relations. A now standard observation [29, 30] is that a tuple $((\mathbb{F}, N, m), (\mathbf{a}, \mathbf{t}))$ is in $\mathbf{R}_{\text{Lookup}}$ if and only if there exists a matrix $M \in \mathbb{F}^{m \times N}$ such that:

- $M \cdot \mathbf{t}^\top = \mathbf{a}^\top$,
- The rows of M are vectors in the standard basis of \mathbb{F}^N . This second condition is equivalent, when $N < \text{char}(\mathbb{F})$, to the equations $M \circ M = M$ (\circ denotes the Hadamard product) and $M \cdot \mathbf{1}^\top = \mathbf{1}^\top$.

It is convenient to translate these last two conditions and express them as relationships between multilinear polynomials, by using the MLEs of the vectors \mathbf{a}, \mathbf{t} and M . A tuple $((\mathbb{F}, N, m), (\mathbf{a}, \mathbf{t}))$ is in $\mathbf{R}_{\text{Lookup}}$ if and only if there exists a $\log(m) + \log(N)$ -variate multilinear polynomial M such that:

- $\sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M(\mathbf{X}, \mathbf{y}) \cdot \mathbf{t}(\mathbf{y}) = \mathbf{a}(\mathbf{X})$
- For all \mathbf{x}, \mathbf{y} , $M(\mathbf{x}, \mathbf{y})^2 = M(\mathbf{x}, \mathbf{y})$; and $\sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M(\mathbf{X}, \mathbf{y}) = 1$. By abuse of notation, we still write the first condition as $M \circ M = M$, and the second one as $M \cdot \mathbf{1}^\top = \mathbf{1}^\top$.

Further, the situation is often such that the Prover commits to the tables \mathbf{a}, \mathbf{t} ³, and has additional witnesses for these commitments. We let $\text{PC} = (\text{Setup}, \text{Commit}, \text{Open}, \text{Eval})$ be a multilinear PCS. For simplicity, we abbreviate Commit as cm . We define the committed matrix algebraic⁴ lookup relation as follows:

$$\mathbf{R}_{\text{CmMAILook}} := \left(\left(\begin{array}{l} (\mathbb{F}, N, m, \mathbf{t}), \\ (\bar{\mathbf{a}}, \bar{M}); \\ (\mathbf{a}, M) \end{array} \right) \middle| \begin{array}{l} N < \text{char}(\mathbb{F}), \\ M \in \mathbb{F}^{\leq 1}[X_1, \dots, X_{\log(m)}, Y_1, \dots, Y_{\log(N)}], \\ \mathbf{a} \in \mathbb{F}^{\leq 1}[X_1, \dots, X_{\log(m)}], \\ \mathbf{t} \in \mathbb{F}^{\leq 1}[Y_1, \dots, Y_{\log(N)}], \\ \bar{M} = \text{cm}(M), \bar{\mathbf{a}} = \text{cm}(\mathbf{a}), \\ \sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M(\mathbf{X}, \mathbf{y}) \cdot \mathbf{t}(\mathbf{y}) = \mathbf{a}(\mathbf{X}), \\ M \circ M = M, M \cdot \mathbf{1}^\top = \mathbf{1}^\top \end{array} \right)$$

This rewriting of the lookup relation will be useful for us when formally describing our folding scheme in Sect. 5.

³ Unless \mathbf{t} has a particular type of structure, e.g., the SOS structure.

⁴ The term “algebraic” refers to the fact that the conditions relating $M, \mathbf{a}, \mathbf{t}$ are expressed with multilinear polynomials.

3.3 SOS-Decomposable Tables

We recall the definition of SOS decomposition from [26].

Definition 3 (SOS decomposition). *Let $c, k \geq 1$, $\alpha = k \cdot c$, and let $\mathbf{t} \in \mathbb{F}^N$ be a table of size N . Assume $N^{1/c}$ is a power of two. Let $\mathbf{t}_1, \dots, \mathbf{t}_\alpha \in \mathbb{F}^{N^{1/c}}$ be α tables of size $N^{1/c}$. We say that \mathbf{t} admits a SOS decomposition with respect to the tables $\mathbf{t}_1, \dots, \mathbf{t}_\alpha$ if there exists a multilinear polynomial $g = g(Y_1, \dots, Y_\alpha) \in \mathbb{F}[Y_1, \dots, Y_\alpha]$ in α variables such that:*

$$\forall \mathbf{y} \in \mathbb{B}^{\log(N)}, \mathbf{t}(\mathbf{y}) = g(\mathbf{t}_1(\mathbf{y}_1), \dots, \mathbf{t}_k(\mathbf{y}_1), \mathbf{t}_{k+1}(\mathbf{y}_2), \dots, \mathbf{t}_\alpha(\mathbf{y}_c))$$

where $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_c) \in (\mathbb{B}^{\log(N)/c})^c$, and further each \mathbf{t}_i can be evaluated in $O(\log(N)/c)$ field operations at any $\mathbf{r} \in \mathbb{F}^{\log(N)/c}$.

3.4 Folding Schemes

We recall the notion of a folding scheme in the sense of [15]. We use the convention that if we say that the protocol has input $(a; b)$, both the Prover and Verifier get a , but only the Prover gets b . The definition is slightly adapted to our case, where the relation and the accumulation relation share the index.

Definition 4. *Fix indexed relations \mathbf{R} and \mathbf{R}_{acc} . An $(\mathbf{R} \rightarrow \mathbf{R}_{\text{acc}})$ -folding scheme is a public-coin interactive protocol \mathcal{P} between a Prover \mathbf{P} and a Verifier \mathbf{V} such that:*

1. *The protocol input is $(\mathfrak{i}, \mathfrak{x}, \mathfrak{x}'; \mathfrak{w}, \mathfrak{w}')$*
2. *When the protocol ends, \mathbf{V} outputs \mathfrak{x}^* , and \mathbf{P} outputs \mathfrak{w}^**
3. **(Perfect) Completeness:** *If $(\mathfrak{i}, \mathfrak{x}; \mathfrak{w}) \in \mathbf{R}_{\text{acc}}$, $(\mathfrak{i}, \mathfrak{x}'; \mathfrak{w}') \in \mathbf{R}$, and \mathbf{P}, \mathbf{V} are honest, then $(\mathfrak{i}, \mathfrak{x}^*; \mathfrak{w}^*) \in \mathbf{R}_{\text{acc}}$ with probability one.*
4. **Knowledge soundness:** *The following protocol $\widetilde{\mathcal{P}}$ between $\widetilde{\mathbf{P}}, \widetilde{\mathbf{V}}$ for the relation $\mathbf{R}_{\text{acc}} \times \mathbf{R}$ is knowledge sound with error negligible in the security parameter:*
 - (a) *Given inputs $(\mathfrak{i}, \mathfrak{x}, \mathfrak{x}'; \mathfrak{w}, \mathfrak{w}')$, $\widetilde{\mathbf{P}}, \widetilde{\mathbf{V}}$ run \mathcal{P} as \mathbf{P}, \mathbf{V} on the same inputs.*
 - (b) *Let $(\mathfrak{i}, \mathfrak{x}^*; \mathfrak{w}^*)$ be the final output of \mathbf{P}, \mathbf{V} in \mathcal{P} . $\widetilde{\mathbf{V}}$ accepts if and only if $(\mathfrak{i}, \mathfrak{x}^*; \mathfrak{w}^*) \in \mathbf{R}_{\text{acc}}$.*

We call instances for the relation \mathbf{R}_{acc} folded or accumulated instances.

This definition allows us to enlarge the relation $\mathbf{R}_{\text{cmMailLook}}$ to a slightly more general one, so that we can apply the folding step. We state the following slight generalization of a lemma in [21] in the sense that it need not be the case that \mathbf{R}_{acc} and \mathbf{R} are identical, but it follows by the same arguments as in [21].

Lemma 1 (Forking Lemma for Folding Schemes, Lemma 1 in [21]). *Consider a $(2\mu + 1)$ -move $(\mathbf{R} \rightarrow \mathbf{R}_{\text{acc}})$ -folding scheme Π . The protocol Π satisfies knowledge soundness if there exists a PPT algorithm Ext such that for all input tuples $(\mathfrak{i}, \mathfrak{x}, \mathfrak{x}')$, outputs witnesses $(\mathfrak{w}, \mathfrak{w}')$ such that $(\mathfrak{i}, \mathfrak{x}; \mathfrak{w}) \in \mathbf{R}_{\text{acc}}$,*

$(\mathbf{i}, \mathbf{x}'; \mathbf{w}') \in \mathbf{R}$; given global parameters \mathbf{gp} and an (n_1, \dots, n_μ) -tree of accepting transcripts and the corresponding folded tuples $(\mathbf{i}, \mathbf{x}^*; \mathbf{w}^*)$. The tree comprises of n_1 transcripts (and the corresponding index instance witness tuples) with fresh randomness in the Verifier's first message ; and for each such transcript, n_2 transcripts (and the corresponding index instance witness tuples) with fresh randomness in the Verifier's second message ; and so on, for a total of $\prod_{i=1}^\mu n_i$ leaves bounded by $\text{poly}(\lambda)$.

4 An IOP and a Folding Scheme for Checking that All Rows in a Matrix Are Elementary Vectors

It is a now standard observation [29,30] that a tuple $((\mathbb{F}, N, m), (\mathbf{a}, \mathbf{t}))$ is in \mathbf{R}_{Look} if and only if there exists a matrix $M \in \mathbb{F}^{m \times N}$ such that:

- $M \cdot \mathbf{t}^\top = \mathbf{a}^\top$,
- The rows of M are vectors in the standard basis of \mathbb{F}^N .

One way to build a folding scheme for lookups is to build a folding scheme for the second condition above. Indeed, if as a result of folding the second condition we combine lookup matrices as $M_1 + \alpha M_2$, then the first condition can also be combined as $(M_1 + \alpha M_2) \cdot \mathbf{t}^\top = (\mathbf{a}_1 + \alpha \mathbf{a}_2)^\top$. By defining an adequate accumulated relation, we can ensure that this is also the case when folding a lookup matrix into an accumulated matrix, one that is the result of potentially many foldings. We let $\text{PC} = (\text{Setup}, \text{Commit}, \text{Open}, \text{Eval})$ be an extractable multilinear PCS. For simplicity, we abbreviate Commit as cm . We want to show that a matrix M seen as a $\log(mN)$ -variate multilinear polynomial is in \mathbf{R}_{elem} :

$$\mathbf{R}_{\text{elem}} := \left\{ ((\mathbb{F}, N, m), \overline{M}; M) \left| \begin{array}{l} m \leq N < \text{char}(\mathbb{F}), \\ M \in \mathbb{F}^{\leq 1}[X_1, \dots, X_{\log(m)}, Y_1, \dots, Y_{\log(N)}], \\ \overline{M} = \text{cm}(M), M \circ M = M, M \cdot \mathbf{1}^\top = \mathbf{1}^\top \end{array} \right. \right\} \quad (6)$$

The notations $M \circ M = M$ and $M \cdot \mathbf{1}^\top = \mathbf{1}^\top$ are shorthand for:

$$\forall (\mathbf{x}, \mathbf{y}) \in \mathbb{B}^{\log(m) + \log(N)}, M(\mathbf{x}, \mathbf{y})^2 - M(\mathbf{x}, \mathbf{y}) = 0 \quad (7)$$

$$\sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M(\mathbf{x}, \mathbf{y}) \equiv 1 \quad (8)$$

These two properties are exactly what we need in order to enforce the fact that each row of M has exactly one 1, and zeros otherwise. Clearly if that is the case, then Eq. (7) and Eq. (8) hold. The converse is also true:

Lemma 2. *Suppose $N < \text{char}(\mathbb{F})$. If Eq. (7) and Eq. (8) hold, then for all $\mathbf{x} \in \mathbb{B}^{\log(m)}$, there exists a unique $\mathbf{y}_{\mathbf{x}} \in \mathbb{B}^{\log(N)}$ such that $M(\mathbf{x}, \mathbf{y}_{\mathbf{x}}) = 1$, and $M(\mathbf{x}, \mathbf{y}) = 0$ for all $\mathbf{y} \in \mathbb{B}^{\log(N)}$ with $\mathbf{y} \neq \mathbf{y}_{\mathbf{x}}$.*

Proof. Eq. (7) implies that for all $(\mathbf{x}, \mathbf{y}) \in \mathbb{B}^{\log(m)+\log(N)}$, $M(\mathbf{x}, \mathbf{y}) = 0$ or $M(\mathbf{x}, \mathbf{y}) = 1$. Then Eq. (8) implies that for all $\mathbf{x} \in \mathbb{B}^{\log(m)}$, $\sum_{\mathbf{y}} M(\mathbf{x}, \mathbf{y}) = 1$. But since $N < \text{char}(\mathbb{F})$, we have that:

$$1 = \sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M(\mathbf{x}, \mathbf{y}) = |\{\mathbf{y} \in \mathbb{B}^{\log(N)} \mid M(\mathbf{x}, \mathbf{y}) = 1\}|$$

■

The PIOP we describe for the relation $\mathbf{R}_{\text{elem}}^5$ is as follows:

1. The Prover sends an oracle $[[M]]$ to a multilinear polynomial supposedly in \mathbf{R}_{elem} .
2. The Verifier samples uniform random elements $\beta \in \mathbb{F}^{\log(mN)}$, $\mathbf{r} \in \mathbb{F}^{\log(m)}$.
3. The Prover and Verifier engage in two sumcheck protocols:

$$\sum_{\mathbf{x} \in \mathbb{B}^{\log(m)}} \sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} (M(\mathbf{x}, \mathbf{y})^2 - M(\mathbf{x}, \mathbf{y})) \cdot \tilde{\text{eq}}(\beta; \mathbf{x}, \mathbf{y}) = 0$$

$$\sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M(\mathbf{r}, \mathbf{y}) = 1$$

4. During these sumchecks, the Verifier runs the sumcheck Verifier on each of the sumchecks. It accepts if the sumcheck Verifier accepts both executions of the sumcheck, and rejects otherwise. Note that in particular, the Verifier queries evaluations of the form $M(\mathbf{r}_1, \mathbf{r}_2)$, $M(\mathbf{r}, \mathbf{r}_3)$, for some random elements $\mathbf{r}_1 \in \mathbb{F}^{\log(m)}$ and $\mathbf{r}_2, \mathbf{r}_3 \in \mathbb{F}^{\log(N)}$ determined during the sumchecks. The Verifier can evaluate $\tilde{\text{eq}}(\beta; \mathbf{r}_1, \mathbf{r}_2)$ on its own.

Lemma 3. *The above protocol is a perfectly complete and knowledge sound PIOP for the relation \mathbf{R}_{elem} , as long as $(4 \log(mN) + 1)/|\mathbb{F}| = \text{negl}(\lambda)$.*

Proof. See the extended version of the paper. ■

4.1 A Folding Scheme for \mathbf{R}_{elem}

We can now construct a folding scheme for \mathbf{R}_{elem} with ideas similar to Nova [21]. Note however, that since our statement is particularly simple we are able to perform a simplification of the cross-term that appears when folding. We let $\text{PC} = (\text{Setup}, \text{Commit}, \text{Open}, \text{Eval})$ be a succinct, binding, extractable, additively homomorphic multilinear PCS. For simplicity, we abbreviate Commit as cm . We define:

$$\mathbf{R}_{\text{elem}}^{\text{acc}} := \left\{ \left(\begin{array}{l} (\mathbb{F}, N, m), \\ (\overline{M}, \overline{E}, \mu); \\ (M, E) \end{array} \right) \left| \begin{array}{l} m \leq N < \text{char}(\mathbb{F}), \\ M, E \in \mathbb{F}^{\leq 1}[X_1, \dots, X_{\log(m)}, Y_1, \dots, Y_{\log(N)}], \mu \in \mathbb{F}, \\ \overline{M} = \text{cm}(M), \overline{E} = \text{cm}(E) \\ M \circ M = M + E, M \cdot \mathbf{1}^T = (1 + \mu) \cdot \mathbf{1}^T \end{array} \right. \right\}$$

⁵ This is slightly abusing notation: the PIOP we describe is for the oracle relation where we replace commitments in \mathbf{R}_{elem} with oracles.

and we now describe a $(\mathbf{R}_{\text{elem}} \rightarrow \mathbf{R}_{\text{elem}}^{\text{acc}})$ -folding scheme, which we call $\mathcal{P}_1 = (\mathbf{P}_1, \mathbf{V}_1)$. We describe it as an interactive protocol, but it can be made non-interactive with the Fiat-Shamir heuristic [16].

Input. The protocol input is $(\overline{M_{\text{acc}}}, \overline{E_{\text{acc}}}, \overline{M}, \mu; M_{\text{acc}}, M)^6$.

1. \mathbf{P}_1 computes and sends the commitment \overline{T} to the cross term: $T = 2(M_{\text{acc}} \circ M) - M$.
2. \mathbf{V}_1 sends uniformly random $\alpha \in \mathbb{F}$.
3. \mathbf{P}_1 and \mathbf{V}_1 output:

$$\overline{M_{\text{acc}}} \leftarrow \overline{M_{\text{acc}}} + \alpha \cdot \overline{M}, \quad \overline{E_{\text{acc}}} \leftarrow \overline{E_{\text{acc}}} + \alpha \cdot \overline{T} + \alpha^2 \cdot \overline{M}, \quad \mu \leftarrow \mu + \alpha.$$

4. \mathbf{P}_1 outputs: $M_{\text{acc}} \leftarrow M_{\text{acc}} + \alpha \cdot M, \quad E_{\text{acc}} \leftarrow E_{\text{acc}} + \alpha \cdot T + \alpha^2 \cdot M$.

Lemma 4. *Protocol \mathcal{P}_1 is a $(\mathbf{R}_{\text{elem}} \rightarrow \mathbf{R}_{\text{elem}}^{\text{acc}})$ -folding scheme which is perfectly complete, and knowledge sound.*

Proof. See the extended version of the paper. ■

Costs We see that in protocol \mathcal{P}_1 :

- \mathbf{P}_1 performs m field doublings, and $8m + 1$ field operations. It also performs 3 group operations, and 3 exponentiations of group elements to the α power. The Prover needs to additionally commit to m field elements (the cross term E).
- \mathbf{V}_1 performs 3 group exponentiations to the α power, 3 group operations, and one field addition.

4.2 A Protocol for Proving Accumulated Instances

To prove a statement of the form $((N, m), \overline{M}, \overline{E}, \mu; M, E) \in \mathbf{R}_{\text{elem}}^{\text{acc}}$, two sum-checks of the form:

$$\sum_{\mathbf{x} \in \mathbb{B}^{\log(m)}} \sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} (M(\mathbf{x}, \mathbf{y})^2 - M(\mathbf{x}, \mathbf{y}) - E(\mathbf{x}, \mathbf{y})) \cdot \tilde{\text{eq}}(\boldsymbol{\beta}; \mathbf{x}, \mathbf{y}) = 0$$

$$\sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M(\mathbf{r}, \mathbf{y}) = 1 + \mu$$

are performed, for random $\boldsymbol{\beta}, \mathbf{r}$ chosen by the Verifier. In particular in the end, the Verifier needs to verify evaluations of the form $M(\mathbf{r}_1, \mathbf{r}_2), E(\mathbf{r}_1, \mathbf{r}_2), M(\mathbf{r}, \mathbf{r}_3)$ for some random $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ determined during the sumchecks. Note that by the updating procedure of M^{acc} and E^{acc} in \mathcal{P}_1 , if we have folded M_0, \dots, M_{n_f} (where initially $M^{\text{acc}} = M_0$), then it holds that at the end of these n_f foldings, the support⁷ of M^{acc} and of E^{acc} is included in the union of the supports of the M_i (equality is possible), in particular they are both at most $n_f \cdot m$ -sparse.

⁶ We use the convention that both Prover and Verifier get what is before the semicolon, and only the Prover gets what is after.

⁷ The support of f is the set of \mathbf{x} such that $f(\mathbf{x}) \neq 0$.

Hence, we may consider that M, E are s -sparse where $s \leq \min\{n_f \cdot m, mN\}$. Provided that we have all the evaluations of $(M(\mathbf{x}, \mathbf{y})^2 - M(\mathbf{x}, \mathbf{y}) - E(\mathbf{x}, \mathbf{y}))$ and of $M(\mathbf{r}, \mathbf{y})$ over their respective hypercubes, [11] shows that we can perform the first sumcheck in $2mN + O(\sqrt{mN})$ field multiplications and the second in N field multiplications. Computing the evaluations of $(M(\mathbf{x}, \mathbf{y})^2 - M(\mathbf{x}, \mathbf{y}) - E(\mathbf{x}, \mathbf{y}))$ over $\mathbb{B}^{\log(mN)}$ takes s field multiplications. Computing the evaluations of $M(\mathbf{r}, \mathbf{y})$ over $\mathbb{B}^{\log(N)}$ can be done in at most $s + m$ field multiplications, which can be seen by first computing the table of values of $\widehat{e}\widehat{q}(\mathbf{x}; \mathbf{r})$ for all $\mathbf{x} \in \mathbb{B}^{\log(m)}$ in m multiplications and writing M in the multilinear Lagrange basis.

Lemma 5. *The protocol above is a perfectly complete, knowledge sound PIOP for the relation $\mathbf{R}_{\text{elem}}^{\text{acc}}$. Suppose M and E are s -sparse (for $s \leq mN$), then in this PIOP the Prover performs at most $2mN + N + 2s + m + O(\sqrt{mN})$ field multiplications, and the Verifier performs $O(\log(mN))$ field operations. The Verifier makes three oracle queries $M(\mathbf{r}_1, \mathbf{r}_2), E(\mathbf{r}_1, \mathbf{r}_2), M(\mathbf{r}, \mathbf{r}_3)$.*

Proof. The proof is very similar to Lemma 3. ■

5 FLI: Folding Lookup Instances

In this section, we describe a method to fold a lookup into a certain relaxed lookup relations that we describe. Starting from the fact that a statement of the form $((\mathbb{F}, N, m), (\mathbf{a}, \mathbf{t})) \in \mathbf{R}_{\text{Lookup}}$ can be expressed as an equation $M \cdot \mathbf{t}^T = \mathbf{a}^T$ where the rows of M are elementary basis vectors, we then use the folding scheme we have developed in Sect. 4. As we mentioned, the random elements used to combine the claims that the matrices M have the correct form will also allow us to fold the claim that $M \cdot \mathbf{t}^T = \mathbf{a}^T$. We let $\text{PC} = (\text{Setup}, \text{Commit}, \text{Open}, \text{Eval})$ be a succinct, binding, extractable, additively homomorphic multilinear PCS. For simplicity, we abbreviate Commit as cm . Start by defining the relaxed relation:

$$\mathbf{R}_1^{\text{acc}} := \left\{ \left(\begin{array}{l} (\mathbb{F}, N, m, \mathbf{t}), \\ (\bar{\mathbf{a}}, \bar{M}, \bar{E}, \mu); \\ (\mathbf{a}, M, E) \end{array} \right) \middle| \begin{array}{l} m \leq N < \text{char}(\mathbb{F}), \\ M, E \in \mathbb{F}^{\leq 1}[X_1, \dots, X_{\log(m)}, Y_1, \dots, Y_{\log(N)}], \\ \mathbf{t} \in \mathbb{F}^{\leq 1}[Y_1, \dots, Y_{\log(N)}], \\ \mathbf{a} \in \mathbb{F}^{\leq 1}[X_1, \dots, X_{\log(m)}], \mu \in \mathbb{F}, \\ \bar{M} = \text{cm}(M), \bar{E} = \text{cm}(E), \bar{\mathbf{a}} = \text{cm}(\mathbf{a}), \\ \sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M(\mathbf{X}, \mathbf{y}) \cdot \mathbf{t}(\mathbf{y}) = \mathbf{a}(\mathbf{X}), \\ M \circ M = M + E, M \cdot \mathbf{1}^T = (1 + \mu) \cdot \mathbf{1}^T \end{array} \right\}$$

This is essentially the accumulated relation in the previous section augmented with the lookup constraint $M \cdot \mathbf{t}^T = \mathbf{a}^T$. By using our folding scheme \mathcal{P}_1 for \mathbf{R}_{elem} , we construct folding schemes for the relation $\mathbf{R}_{\text{cmMAILook}}$ into $\mathbf{R}_1^{\text{acc}}$.

5.1 FLI: a $(\mathbf{R}_{\text{CmMAILook}} \rightarrow \mathbf{R}_1^{\text{acc}})$ -Folding Scheme

The folding scheme is described as an interactive protocol, but it can be made non-interactive with the Fiat-Shamir heuristic [16]. We denote this folding scheme by $\mathcal{F}_1 = (\tilde{\mathcal{P}}_1, \tilde{\mathcal{V}}_1)$.

Input. The protocol input is⁸

$$((N, m), \mathbf{t}, \overline{\mathbf{a}}_{\text{acc}}, \overline{M}_{\text{acc}}, \overline{E}_{\text{acc}}, \mu, \overline{M}, \overline{\mathbf{a}}; \mathbf{a}_{\text{acc}}, M_{\text{acc}}, E_{\text{acc}}, \mathbf{a}, M).$$

1. $\tilde{\mathcal{P}}_1$ and $\tilde{\mathcal{V}}_1$ follow protocol \mathcal{P}_1 with input

$$((N, m), \overline{M}_{\text{acc}}, \overline{E}_{\text{acc}}, \overline{M}, \mu; M_{\text{acc}}, E_{\text{acc}}, M).$$

At the end of \mathcal{P}_1 , $\tilde{\mathcal{P}}_1$ and $\tilde{\mathcal{V}}_1$ output:

$$\overline{M}_{\text{acc}} \leftarrow \overline{M}_{\text{acc}} + \alpha \cdot \overline{M}, \quad \overline{E}_{\text{acc}} \leftarrow \overline{E}_{\text{acc}} + \alpha \cdot \overline{T} + \alpha^2 \cdot \overline{M}, \quad \mu \leftarrow \mu + \alpha$$

while $\tilde{\mathcal{P}}_1$ outputs:

$$M_{\text{acc}} \leftarrow M_{\text{acc}} + \alpha \cdot M, \quad E_{\text{acc}} \leftarrow E_{\text{acc}} + \alpha \cdot T + \alpha^2 M$$

for some random element $\alpha \in \mathbb{F}$ determined during the course of \mathcal{P}_1 , and $T := 2(M_{\text{acc}} \circ M) - M$.

2. $\tilde{\mathcal{P}}_1$ and $\tilde{\mathcal{V}}_1$ output: $\overline{\mathbf{a}}_{\text{acc}} \leftarrow \overline{\mathbf{a}}_{\text{acc}} + \alpha \cdot \overline{\mathbf{a}}$; and $\tilde{\mathcal{P}}_1$ outputs: $\mathbf{a}_{\text{acc}} \leftarrow \mathbf{a}_{\text{acc}} + \alpha \cdot \mathbf{a}$.

Lemma 6. *Protocol \mathcal{F}_1 is a $(\mathbf{R}_{\text{CmMAILook}} \rightarrow \mathbf{R}_1^{\text{acc}})$ -folding scheme that is perfectly complete, and knowledge sound.*

Proof. See the extended version of the paper. ■

Costs

We see that the Prover and Verifier work in \mathcal{F}_1 is almost the same as in \mathcal{P}_1 . The additional work is: one group exponentiation, one group multiplication for both the Prover and Verifier. Further, the Prover makes $2m$ extra field operations.

5.2 A Protocol for Proving Accumulated Instances.

To prove a statement of the form $((N, m), \mathbf{t}, \overline{\mathbf{a}}, \overline{M}, \overline{E}, \mu; \mathbf{a}, M, E) \in \mathbf{R}_1^{\text{acc}}$, we use the same protocol that proves accumulated instances for \mathcal{P}_1 (Sect. 4.2), together with an additional sumcheck of the form:

$$\sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M(\mathbf{r}, \mathbf{y}) \cdot \mathbf{t}(\mathbf{y}) = \mathbf{a}(\mathbf{r})$$

for some random \mathbf{r} . Again, we may assume M, E are s -sparse with $s \leq \min\{n_f \cdot m, mN\}$ (n_f the number of folding steps, see Sect. 4.2). Using [11],

⁸ Still with the convention that both Prover and Verifier get what is before the semi-colon, and only the Prover gets what is after.

if we have all evaluations of $M(\mathbf{r}, \mathbf{y})\mathbf{t}(\mathbf{y})$ over $\mathbb{B}^{\log(N)}$, this sumcheck costs $5N$ field multiplications for the Prover. Computing all evaluations of $M(\mathbf{r}, \mathbf{y})\mathbf{t}(\mathbf{y})$ can be done in $2s + m$ field multiplications. We can also save costs by batching this sumcheck with the sumcheck of the form $\sum_{\mathbf{y}} M(\mathbf{r}', \mathbf{y}) = (1 + \mu)$ from the protocol that proves accumulated instances for \mathcal{P}_1 . The Verifier samples a single random element $\mathbf{r} \in \mathbb{F}^{\log(m)}$ for both of these sumchecks, and an additional combination random element $\gamma \in \mathbb{F}$. All in all, the protocol performs the two following sumchecks:

$$\sum_{\mathbf{x} \in \mathbb{B}^{\log(m)}} \sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} (M(\mathbf{x}, \mathbf{y})^2 - M(\mathbf{x}, \mathbf{y}) - E(\mathbf{x}, \mathbf{y})) \cdot \tilde{\text{eq}}(\beta; \mathbf{x}, \mathbf{y}) = 0$$

$$\sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M(\mathbf{r}, \mathbf{y}) + \gamma \cdot M(\mathbf{r}, \mathbf{y}) \cdot \mathbf{t}(\mathbf{y}) = 1 + \mu + \gamma \cdot \mathbf{a}(\mathbf{r}),$$

and reduces to the evaluations $M(\mathbf{r}_1, \mathbf{r}_2), E(\mathbf{r}_1, \mathbf{r}_2), M(\mathbf{r}, \mathbf{r}_3), \mathbf{a}(\mathbf{r})$ for the random elements $\mathbf{r}_1 \in \mathbb{F}^{\log(m)}$ and $\mathbf{r}_2, \mathbf{r}_3 \in \mathbb{F}^{\log(N)}$ determined during the sumchecks.

Lemma 7. *The protocol above is a perfectly complete and knowledge sound PIOP for the relation $\mathbf{R}_1^{\text{acc}}$. Suppose M and E are s -sparse (for $s \leq mN$), then in this PIOP the Prover performs*

$$2mN + N + 3s + m + O(\sqrt{mN})$$

field multiplications, and the Verifier performs $O(\log(mN))$ field operations. The Verifier makes four oracle queries $M(\mathbf{r}_1, \mathbf{r}_2), E(\mathbf{r}_1, \mathbf{r}_2), M(\mathbf{r}, \mathbf{r}_3), \mathbf{a}(\mathbf{r})$. ■

5.3 Extending SOS Decompositions for Folding

In Lasso [26], when \mathbf{t} is SOS-decomposable (into $\alpha = k \cdot c$ tables $\mathbf{t}_1, \dots, \mathbf{t}_\alpha$ and polynomial g , see Definition 3), one needs to verify that for all $\mathbf{x} \in \mathbb{B}^{\log(m)}$ the following condition holds:

$$\mathbf{a}(\mathbf{x}) = g(\mathbf{E}_1(\mathbf{x}), \dots, \mathbf{E}_\alpha(\mathbf{x})) \wedge \exists \mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_c) \in \left(\mathbb{B}^{\log(N)/c}\right)^c, \mathbf{E}_i(\mathbf{x}) = \mathbf{t}_i(\mathbf{y}_{\lceil i/k \rceil})$$

where the \mathbf{E}_i are vectors of length m . The first of these conditions is verified with a sumcheck, and Lasso uses an offline memory-checking technique to certify that the second condition holds. Note however, that since the \mathbf{E}_i are included in the \mathbf{t}_i , their entries could a priori be arbitrary elements of \mathbb{F}^9 . As is remarked in Lasso, this can represent a significant overhead when committing to the vectors \mathbf{E}_i by using curve-based schemes.

⁹ This is mitigated by the fact that in tables that arise in practice, for example in the RISC-V instruction set, the entries in the tables appearing in the SOS decompositions are relatively small.

As we remarked when discussing Eq. (4), we can express the same conditions in a different way. When \mathbf{t} is SOS-decomposable, the fact that for all $\mathbf{x} \in \mathbb{B}^{\log(m)}$ there exists $\mathbf{y} \in \mathbb{B}^{\log(N)}$ such that $\mathbf{a}(\mathbf{x}) = \mathbf{t}(\mathbf{y})$ can be expressed as:

$$\forall \mathbf{x} \in \mathbb{B}^{\log(m)}, \exists \mathbf{y} \in \mathbb{B}^n, \mathbf{a}(\mathbf{x}) = \mathbf{t}(\mathbf{y}) = g(\mathbf{t}_1(\mathbf{y}_1), \dots, \mathbf{t}_k(\mathbf{y}_1), \mathbf{t}_{k+1}(\mathbf{y}_2), \dots, \mathbf{t}_\alpha(\mathbf{y}_c)) \quad (9)$$

So we can think of Eq. (9) as needing to point, for each $\mathbf{x} \in \mathbb{B}^{\log(m)}$, to the correct indices of the tables $\mathbf{t}_1, \dots, \mathbf{t}_\alpha$ that make the equation hold. Note that the \mathbf{t}_i have size $N^{1/c}$, so we know that we can point to the correct indices for all $\mathbf{x} \in \mathbb{B}^{\log(m)}$ by using matrices of size $m \times N^{1/c}$ that have elementary basis vectors as rows. Therefore Eq. (9) is equivalent to:

$$\forall \mathbf{x} \in \mathbb{B}^{\log(m)}, \mathbf{a}(\mathbf{x}) = g(M_1(\mathbf{x})\mathbf{t}_1, \dots, M_1(\mathbf{x})\mathbf{t}_k, M_2(\mathbf{x})\mathbf{t}_{k+1}, \dots, M_c(\mathbf{x})\mathbf{t}_\alpha) \quad (10)$$

for some matrices M_1, \dots, M_c of size $m \times N^{1/c}$ which have the special form we have been talking about: *each of their rows is a vector in the standard basis of $\mathbb{F}^{N^{1/c}}$* . Importantly, committing to the M_i with curve-based commitment schemes consists only in group *operations*. As we mentioned, one should think as the matrices M_i (for $i \in [c]$) as pointing to the correct entries of $\mathbf{t}_{(i-1)k+1}, \dots, \mathbf{t}_{ik}$ such that Eq. (9) holds. For simplicity we use the shorthand:

$$\forall \mathbf{x} \in \mathbb{B}^{\log(m)}, \forall i \in [c], (i-1)k+1 \leq j \leq ik, M_i(\mathbf{x})\mathbf{t}_j := \sum_{\mathbf{y} \in \mathbb{B}^n} M_i(\mathbf{x}, \mathbf{y}) \cdot \mathbf{t}_j(\mathbf{y}) \quad (11)$$

Note that the $M_i(\mathbf{X})\mathbf{t}_j$ are $\log(m)$ -variate multilinear polynomials whose evaluations over the hypercube can be computed by simply selecting entries from \mathbf{t}_j . We can verify Eq. (10) with a sumcheck of the form:

$$\sum_{\mathbf{x} \in \mathbb{B}^{\log(m)}} (\mathbf{a}(\mathbf{x}) - g(M_1(\mathbf{x})\mathbf{t}_1, \dots, M_1(\mathbf{x})\mathbf{t}_k, M_2(\mathbf{x})\mathbf{t}_{k+1}, \dots, M_c(\mathbf{x})\mathbf{t}_\alpha)) \cdot \tilde{\mathbf{e}}\mathbf{q}(\boldsymbol{\beta}; \mathbf{x}) = 0$$

for a random $\boldsymbol{\beta} \in \mathbb{F}^{\log(m)}$, which will reduce to claims of the form $\mathbf{a}(\mathbf{r}) = d, M_i(\mathbf{r})\mathbf{t}_j = v_{i,j}$ for some random $\mathbf{r} \in \mathbb{F}^{\log(m)}$ and $d, v_{i,j} \in \mathbb{F}$ (for $i \in [c]$ and $(i-1)k+1 \leq j \leq ik$). This will allow us to use our folding scheme for the relation $\mathbf{R}_{\text{CmMAILook}}$ to fold these claims about the evaluations of the $M_i(\mathbf{r})\mathbf{t}_j = v_{i,j}$, as per the following remark.

Remark 1. (The $M \cdot \mathbf{t}^\top = \mathbf{a}^\top$ condition) In Sect. 5.1 we described a folding scheme for the relation $\mathbf{R}_{\text{CmMAILook}}$, in which the condition relating M , \mathbf{t} and \mathbf{a} is:

$$\sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M(\mathbf{X}, \mathbf{y}) \cdot \mathbf{t}(\mathbf{y}) = \mathbf{a}(\mathbf{X})$$

We could equally have described the folding scheme for a "randomized" version of this condition, of the form:

$$\sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M(\mathbf{r}, \mathbf{y}) \cdot \mathbf{t}(\mathbf{y}) = \mathbf{a}(\mathbf{r})$$

for some random $\mathbf{r} \in \mathbb{F}^{\log(m)}$. The two folding schemes we have described translate mutatis mutandis to this setting, *provided that the random evaluation vector r is the same for the accumulated claim and the new claim*. We do not write this again as it is quite literally the same folding scheme. The protocol that proves accumulated instances of the non-randomized versions of our folding schemes starts precisely by randomizing the condition $\sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M^{\text{acc}}(\mathbf{X}, \mathbf{y}) \cdot \mathbf{t}(\mathbf{y}) = \mathbf{a}^{\text{acc}}(\mathbf{X})$ as $\sum_{\mathbf{y} \in \mathbb{B}^{\log(N)}} M^{\text{acc}}(\mathbf{r}, \mathbf{y}) \cdot \mathbf{t}(\mathbf{y}) = \mathbf{a}^{\text{acc}}(\mathbf{r})$. We still refer to the folding scheme for this randomized condition $M \cdot \mathbf{t}^\top = \mathbf{a}^\top$ as \mathcal{F}_1 . The soundness loss of this randomization is the probability that the randomized condition holds while the condition on polynomials does not. This has probability at most $\log(m)/|\mathbb{F}|$ by Schwartz-Zippel, since all polynomials are multilinear. This will be useful in the next section, when we use SOS decompositions in conjunction with the folding schemes we have constructed.

5.4 FLI + SOS

In this section, we combine SOS decompositions with FLI using our remarks from Sect. 5.3: by modifying the SOS decomposition step from the Lasso paper [26] we make it such that the Prover only needs to commit to sparse binary matrices. Recall from the previous section that when we are looking up \mathbf{a} into an SOS-decomposable table \mathbf{t} we can express the lookup condition as a sumcheck in $\log(m)$ variables:

$$\sum_{\mathbf{x} \in \mathbb{B}^{\log(m)}} (\mathbf{a}(\mathbf{x}) - g(M_1(\mathbf{x})\mathbf{t}_1, \dots, M_1(\mathbf{x})\mathbf{t}_k, M_2(\mathbf{x})\mathbf{t}_{k+1}, \dots, M_c(\mathbf{x})\mathbf{t}_\alpha)) \cdot \tilde{\text{eq}}(\boldsymbol{\beta}; \mathbf{x}) = 0$$

for matrices M_1, \dots, M_c of size $m \times N^{1/c}$ in \mathbf{R}_{elem} , and a random $\boldsymbol{\beta} \in \mathbb{F}^{\log(m)}$. The sumcheck reduces to claims of the form $\mathbf{a}(\mathbf{r}) = d, M_i(\mathbf{r})\mathbf{t}_j = v_{i,j}$ for some random $\mathbf{r} \in \mathbb{F}^{\log(m)}$ and $d, v_{i,j} \in \mathbb{F}$. We can now fold the claims about the matrices being in \mathbf{R}_{elem} using Protocol \mathcal{P}_1 , and in parallel fold the claims about the evaluations. The only important detail (see Remark 1) is that the random evaluation point is the same, which we can always enforce with a technique we call the *point-shifting sumcheck*. This is a quite standard technique (see for example [20]), that allows to reduce the evaluation of two (or more) multilinear polynomials at different point to evaluations at the same point. It exploits that for any multilinear polynomial $f \in \mathbb{F}^{\leq 1}[X_1, \dots, X_s]$ and any $\mathbf{r} \in \mathbb{F}^s$, it holds that $f(\mathbf{r}) = \sum_{\mathbf{x} \in \mathbb{B}^s} f(\mathbf{x}) \cdot \tilde{\text{eq}}(\mathbf{x}; \mathbf{r})$ (see Eq. (5)). Say we have two multilinear polynomials f, g in s variables with purported evaluations $f(\mathbf{r}_1) = c, g(\mathbf{r}_2) = d$ respectively (for some $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{F}^s$), we may apply the sumcheck protocol to certify that the following holds:

$$\sum_{\mathbf{x} \in \mathbb{B}^s} f(\mathbf{x}) \cdot \tilde{\text{eq}}(\mathbf{x}; \mathbf{r}_1) + \gamma \cdot g(\mathbf{x}) \cdot \tilde{\text{eq}}(\mathbf{x}; \mathbf{r}_2) = c + \gamma \cdot d$$

for a random uniform $\gamma \in \mathbb{F}$. At the end of this sumcheck, the Prover needs to reveal $f(\mathbf{r}), g(\mathbf{r})$ and $\tilde{\text{eq}}(\mathbf{r}; \mathbf{r}_1), \tilde{\text{eq}}(\mathbf{r}; \mathbf{r}_2)$. In this way, we now have reduced the

statements that $f(\mathbf{r}_1) = c, g(\mathbf{r}_2) = d$ to an evaluation of f and g at the same point.

Formally, we will obtain a folding scheme from the committed matrix lookup SOS relation:

$$\mathbf{R}^{\text{SOS}} := \left\{ \begin{array}{l} \left(\begin{array}{l} (\mathbb{F}, N, m, \mathbf{t}, \alpha, \\ c, k, \mathbf{t}_1, \dots, \mathbf{t}_\alpha, g), \\ (\overline{M}_1, \dots, \overline{M}_c, \overline{\mathbf{a}}); \\ (M_1, \dots, M_c, \mathbf{a}) \end{array} \right) \\ \left. \begin{array}{l} N < \text{char}(\mathbb{F}), \alpha = k \cdot c \in \mathbb{N}, \\ \forall i \in [c], M_i \in \mathbb{F}^{\leq 1}[\mathbf{X}_{\lfloor \log(m) \rfloor}], \mathbf{Y}_{\lfloor \log(N)/c \rfloor}, \\ \mathbf{a} \in \mathbb{F}^{\leq 1}[\mathbf{X}_{\lfloor \log(m) \rfloor}], \mathbf{t} \in \mathbb{F}^{\leq 1}[\mathbf{Y}_{\lfloor \log(N)/c \rfloor}], \\ \mathbf{t}_1, \dots, \mathbf{t}_\alpha \in \mathbb{F}^{\leq 1}[\mathbf{Y}_{\lfloor \log(N)/c \rfloor}], \\ \forall i \in [c], \overline{M}_i = \text{cm}(M_i), \overline{\mathbf{a}} = \text{cm}(\mathbf{a}), \\ \forall \mathbf{y} \in \mathbb{B}^{\log(N)}, \mathbf{t}(\mathbf{y}) = g(\mathbf{t}_1(\mathbf{y}_1), \dots, \mathbf{t}_\alpha(\mathbf{y}_c)), \\ \forall \mathbf{x} \in \mathbb{B}^{\log(m)}, \mathbf{a}(\mathbf{x}) = g(M_1(\mathbf{x})\mathbf{t}_1, \dots, M_c(\mathbf{x})\mathbf{t}_\alpha), \\ \forall i \in [c], M_i \circ M_i = M_i, M_i \cdot \mathbf{1}^\top = \mathbf{1}^\top \end{array} \right\}$$

(for brevity $\mathbf{X}_{\lfloor \log(m) \rfloor} = (X_1, \dots, X_{\lfloor \log(m) \rfloor})$, $\mathbf{Y}_{\lfloor \log(N)/c \rfloor} = (Y_1, \dots, Y_{\lfloor \log(N)/c \rfloor})$) into the accumulated relation:

$$\mathbf{R}^{\text{accSOS}} := \left\{ \begin{array}{l} \left(\begin{array}{l} (\mathbb{F}, N, m, \mathbf{t}, \alpha, \\ c, k, \mathbf{t}_1, \dots, \mathbf{t}_\alpha, g), \\ (\overline{M}, \overline{M}_1, \dots, \overline{M}_c, \overline{E}, \overline{\mathbf{a}}, \mathbf{r}, \\ d, \mu, c_1, \dots, c_\alpha); \\ (M, M_1, \dots, M_c, E, \mathbf{a}) \end{array} \right) \\ \left. \begin{array}{l} N < \text{char}(\mathbb{F}), \alpha = k \cdot c \in \mathbb{N}, \\ \mathbf{r} \in \mathbb{F}^{\log(m)}, d, c_1, \dots, c_\alpha, \mu \in \mathbb{F}, \\ M, M_1, \dots, M_c, E \in \mathbb{F}^{\leq 1}[\mathbf{X}_{\lfloor \log(m) \rfloor}], \mathbf{Y}_{\lfloor \log(N)/c \rfloor}, \\ \mathbf{a} \in \mathbb{F}^{\leq 1}[\mathbf{X}_{\lfloor \log(m) \rfloor}], \mathbf{t} \in \mathbb{F}^{\leq 1}[\mathbf{Y}_{\lfloor \log(N)/c \rfloor}], \\ \mathbf{t}_1, \dots, \mathbf{t}_\alpha \in \mathbb{F}^{\leq 1}[\mathbf{Y}_{\lfloor \log(N)/c \rfloor}], \\ \overline{M} = \text{cm}(M), \overline{E} = \text{cm}(E), \overline{\mathbf{a}} = \text{cm}(\mathbf{a}), \\ \forall i \in [c], \overline{M}_i = \text{cm}(M_i), \\ \forall \mathbf{y} \in \mathbb{B}^{\log(N)}, \mathbf{t}(\mathbf{y}) = g(\mathbf{t}_1(\mathbf{y}_1), \dots, \mathbf{t}_\alpha(\mathbf{y}_c)), \\ \mathbf{a}(\mathbf{r}) = d, M_1(\mathbf{r})\mathbf{t}_1 = c_1, \dots, M_c(\mathbf{r})\mathbf{t}_\alpha = c_\alpha, \\ M \circ M = M + E, M \cdot \mathbf{1}^\top = (1 + \mu) \cdot \mathbf{1}^\top \end{array} \right\}$$

We describe this next as an interactive protocol, but it can be made non-interactive with the Fiat-Shamir heuristic [16]. We denote it by $\mathcal{F}_1^{\text{SOS}} = (\mathcal{P}_1^{\text{SOS}}, \mathcal{V}_1^{\text{SOS}})$.

Input. The Protocol input is

$$\left(\begin{array}{l} (\mathbb{F}, N, m), \alpha, c, k, \mathbf{t}, \mathbf{t}_1, \dots, \mathbf{t}_\alpha, g, \overline{M}^{\text{acc}}, \overline{M}_1^{\text{acc}}, \dots, \\ \overline{M}_c^{\text{acc}}, \overline{E}^{\text{acc}}, \overline{\mathbf{a}}^{\text{acc}}, \mathbf{r}^{\text{acc}}, \mu, d^{\text{acc}}, c_1^{\text{acc}}, \dots, c_\alpha^{\text{acc}}, \overline{M}_1, \dots, \overline{M}_c, \overline{\mathbf{a}}; \\ M^{\text{acc}}, M_1^{\text{acc}}, \dots, M_c^{\text{acc}}, E^{\text{acc}}, \mathbf{a}^{\text{acc}}, M_1, \dots, M_c, \mathbf{a} \end{array} \right)$$

1. $\mathsf{P}_1^{\text{SOS}}$ and $\mathsf{V}_1^{\text{SOS}}$ engage in a sumcheck of the form:

$$\sum_{\mathbf{x} \in \mathbb{F}^{\log(m)}} \phi(\mathbf{x}) = \gamma \cdot c_1^{\text{acc}} + \dots + \gamma^{\alpha+1} \cdot c_\alpha^{\text{acc}} + \gamma^{\alpha+2} \cdot d^{\text{acc}}$$

where

$$\begin{aligned} \phi(\mathbf{x}) = & (\mathbf{a}(\mathbf{x}) - g(M_1(\mathbf{x})\mathbf{t}_1, \dots, M_c(\mathbf{x})\mathbf{t}_\alpha)) \cdot \tilde{\mathbf{e}}\mathbf{q}(\boldsymbol{\beta}, \mathbf{x}) \\ & + \gamma \cdot M_1^{\text{acc}}(\mathbf{x})\mathbf{t}_1 \cdot \tilde{\mathbf{e}}\mathbf{q}(\mathbf{r}^{\text{acc}}, \mathbf{x}) + \dots + \gamma^{\alpha+1} \cdot M_c^{\text{acc}}(\mathbf{x})\mathbf{t}_\alpha \cdot \tilde{\mathbf{e}}\mathbf{q}(\mathbf{r}^{\text{acc}}, \mathbf{x}) \\ & + \gamma^{\alpha+2} \cdot \mathbf{a}^{\text{acc}}(\mathbf{x}) \cdot \tilde{\mathbf{e}}\mathbf{q}(\mathbf{r}^{\text{acc}}, \mathbf{x}) \end{aligned}$$

for some random uniform $\boldsymbol{\beta} \in \mathbb{F}^{\log(m)}$ and $\gamma \in \mathbb{F}$ chosen by $\mathsf{V}_1^{\text{SOS}}$. At the end of the sumcheck, $\mathsf{V}_1^{\text{SOS}}$ needs to check a single equation involving the evaluations

$$\begin{aligned} & \tilde{\mathbf{e}}\mathbf{q}(\boldsymbol{\beta}, \mathbf{r}), \tilde{\mathbf{e}}\mathbf{q}(\mathbf{r}^{\text{acc}}, \mathbf{r}), \mathbf{a}(\mathbf{r}), \mathbf{a}^{\text{acc}}(\mathbf{r}), \\ & M_1(\mathbf{r})\mathbf{t}_1, \dots, M_c(\mathbf{r})\mathbf{t}_\alpha, M_1^{\text{acc}}(\mathbf{r})\mathbf{t}_1, \dots, M_c^{\text{acc}}(\mathbf{r})\mathbf{t}_\alpha, \end{aligned}$$

where the random challenge $\mathbf{r} \in \mathbb{F}^{\log(m)}$ is determined during the course of the sumcheck. It evaluates the first two by itself.

2. Now, for $i = 1$ to $i = c$:

– $\mathsf{P}_1^{\text{SOS}}$ computes and sends the commitments \overline{T}_i to the cross term:

$$T_i := 2(M^{\text{acc}} \circ M_i) - M_i$$

– $\mathsf{V}_1^{\text{SOS}}$ chooses a uniform random element $\eta_i \in \mathbb{F}$. $\mathsf{P}_1^{\text{SOS}}$ and $\mathsf{V}_1^{\text{SOS}}$ output:

$$\begin{aligned} \overline{M}^{\text{acc}} & \leftarrow \overline{M}^{\text{acc}} + \eta_i \cdot \overline{M}_i, & \overline{E}^{\text{acc}} & \leftarrow \overline{E}^{\text{acc}} + \eta_i \cdot \overline{T}_i + \eta_i^2 \cdot \overline{M}_i \\ \mu & \leftarrow \mu + \eta_i, & \overline{M}_i^{\text{acc}} & \leftarrow \overline{M}_i^{\text{acc}} + \eta_i \cdot \overline{M}_i \end{aligned}$$

and for all $j \in \{k(i-1) + 1, \dots, ki\}$, $c_j^{\text{acc}} \leftarrow M_i^{\text{acc}}(\mathbf{r})\mathbf{t}_j + \eta_i \cdot M_i(\mathbf{r})\mathbf{t}_j$.

– $\mathsf{P}_1^{\text{SOS}}$ outputs:

$$\begin{aligned} M^{\text{acc}} & \leftarrow M^{\text{acc}} + \eta_i \cdot M_i, & E^{\text{acc}} & \leftarrow E^{\text{acc}} + \eta_i \cdot T_i + \eta_i^2 \cdot M_i \\ & & M_i^{\text{acc}} & \leftarrow M_i^{\text{acc}} + \eta_i \cdot M_i \end{aligned}$$

3. $\mathsf{V}_1^{\text{SOS}}$ chooses uniform random element $\theta \in \mathbb{F}$. $\mathsf{P}_1^{\text{SOS}}$ and $\mathsf{V}_1^{\text{SOS}}$ output:

$$\overline{\mathbf{a}}^{\text{acc}} \leftarrow \overline{\mathbf{a}}^{\text{acc}} + \theta \cdot \overline{\mathbf{a}}, \quad \mathbf{r}^{\text{acc}} \leftarrow \mathbf{r}, \quad d^{\text{acc}} \leftarrow \mathbf{a}^{\text{acc}}(\mathbf{r}) + \theta \cdot \mathbf{a}(\mathbf{r})$$

4. $\mathsf{P}_1^{\text{SOS}}$ outputs: $\mathbf{a}^{\text{acc}} \leftarrow \mathbf{a}^{\text{acc}} + \theta \cdot \mathbf{a}$

Lemma 8. *Protocol $\mathcal{F}_1^{\text{SOS}}$ is a $(\mathbf{R}^{\text{SOS}} \rightarrow \mathbf{R}_1^{\text{accSOS}})$ -folding scheme that is perfectly complete, and knowledge sound.*

Proof. See the extended version of the paper. ■

Costs. All in all, we see that in protocol $\mathcal{F}_1^{\text{SOS}}$:

- The Prover needs to perform the initial sumcheck which costs $m \cdot (\max(\deg(g), 2) + 1) \cdot (6\alpha + |g| + 12)$ field operations using the formula from [17]. The Prover needs to perform $4c + 1$ group exponentiations, and $4c + 1$ group additions. The Prover also needs to perform $O(\alpha m)$ field operations, where the constant is small.
- The Verifier needs to perform $O(\alpha \log(m))$ field operations in the sumcheck, $4c + 1$ group exponentiations, and $4c + 1$ group additions. It also needs to perform $2\alpha + 2$ field operations.

Proving Accumulated Instances. To prove accumulated instances, we need to verify the following:

$$\begin{aligned} \mathbf{a}(\mathbf{r}) &= d, M_1(\mathbf{r})\mathbf{t}_1 = c_1, \dots, M_c(\mathbf{r})\mathbf{t}_\alpha = c_\alpha \\ M \circ M &= M + E, M \cdot \mathbf{1}^\top = (1 + \mu) \cdot \mathbf{1}^\top \end{aligned}$$

The first claim is proved with an evaluation query to \mathbf{a} . The second and last claim are solved with a single sumcheck of the form:

$$\begin{aligned} \sum_{\mathbf{y} \in \mathbb{B}^{\log(N)/c}} M(\mathbf{r}', \mathbf{y}) + \delta \cdot M_1(\mathbf{r}, \mathbf{y}) \cdot \mathbf{t}_1(\mathbf{y}) + \delta^\alpha \cdot M_c(\mathbf{r}, \mathbf{y}) \cdot \mathbf{t}_\alpha(\mathbf{y}) = \\ (1 + \mu) + \delta \cdot c_1 + \dots + \delta^\alpha \cdot c_\alpha \end{aligned}$$

for a randomly sampled $\delta \in \mathbb{F}$ and $\mathbf{r}' \in \mathbb{F}^{\log(m)}$. Similarly to Sect. 4.2 and Sect. 5.2, we may assume that the M, M_i, E are s -sparse ($s \leq mN^{1/c}$). To apply the techniques in [11], we need to compute the evaluations $M(\mathbf{r}', \mathbf{y})$ and $\delta \cdot M_1(\mathbf{r}, \mathbf{y}) \cdot \mathbf{t}_1(\mathbf{y}), \dots, \delta^\alpha \cdot M_c(\mathbf{r}, \mathbf{y}) \cdot \mathbf{t}_\alpha(\mathbf{y})$ over $\mathbb{B}^{\log(N)/c}$. This can be done in no more than $s + m$ field multiplications for $M'(\mathbf{r}', \mathbf{y})$, and $3\alpha s + m$ field multiplications for all the other products. Hence, the Prover in this sumcheck does no more than $(5\alpha + 1)N^{1/c} + (3\alpha + 1)s + 2m$ field multiplications. At the end of the sumcheck the Prover needs to reveal evaluations $M_1(\mathbf{r}, \mathbf{r}_1), \dots, M_c(\mathbf{r}, \mathbf{r}_1), M(\mathbf{r}', \mathbf{r}_1), \mathbf{t}_1(\mathbf{r}_1), \dots, \mathbf{t}_\alpha(\mathbf{r}_1)$ for $\mathbf{r}_1 \in \mathbb{F}^{\log(N)/c}$ determined during the sumcheck. By the SOS assumption the Verifier can evaluate the \mathbf{t}_i 's in $O(\alpha \log(N)/c)$ field operations. The remaining condition is also checked with a single sumcheck of the form:

$$\sum_{\mathbf{x} \in \mathbb{B}^{\log(m)}} \sum_{\mathbf{y} \in \mathbb{B}^{\log(N)/c}} ((M(\mathbf{x}, \mathbf{y}))^2 - M(\mathbf{x}, \mathbf{y}) - E(\mathbf{x}, \mathbf{y})) \cdot \tilde{\text{eq}}(\beta; \mathbf{x}, \mathbf{y}) = 0$$

for a randomly sampled β . We already saw that in this sumcheck the Prover performs no more than $2mN^{1/c} + s + O(\sqrt{mN^{1/c}})$ field multiplications. At the end of this sumcheck, the Prover needs to reveal evaluations $M(\mathbf{r}_2, \mathbf{r}_3), E(\mathbf{r}_2, \mathbf{r}_3), \tilde{\text{eq}}(\beta; \mathbf{r}_2, \mathbf{r}_3)$ for $\mathbf{r}_2 \in \mathbb{F}^{\log(m)}, \mathbf{r}_3 \in \mathbb{F}^{\log(N)/c}$ determined during the sumcheck. The Verifier can evaluate $\tilde{\text{eq}}(\beta; \mathbf{r}_2, \mathbf{r}_3)$ in $O(\log(m) + \log(N)/c)$ field operations.

Lemma 9. *The protocol Π_1^{accSOS} we just described is a perfectly complete and knowledge sound PIOP for the relation $\mathbf{R}_1^{\text{accSOS}}$. Suppose all M_i, M, E are s -sparse (for $s \leq mN^{1/c}$), then in this PIOP the Prover performs no more than:*

$$2mN^{1/c} + (5\alpha + 1)N^{1/c} + (3\alpha + 2)s + 2m + O(\sqrt{mN^{1/c}})$$

field multiplications, and the Verifier performs $O(\log(mN^{1/c}))$ field operations. The Verifier makes $c + 4$ oracle queries $M_1(\mathbf{r}, \mathbf{r}_1), \dots, M_c(\mathbf{r}, \mathbf{r}_1), M(\mathbf{r}', \mathbf{r}_1), M(\mathbf{r}_2, \mathbf{r}_3), E(\mathbf{r}_2, \mathbf{r}_3), \mathbf{a}(\mathbf{r})$. ■

6 Performance

In this section we discuss performance aspects of FLI+SOS. First, it is clear from Tables 2 and 3 that FLI+SOS has the most efficient folding Prover and Verifier compared to Protostar+SOS and DeepThought+SOS current state-of-the-art schemes. When it comes to proving accumulated instances, FLI+SOS’s Prover incurs a cost with a term of the form $2mN^{1/c}$. While this is prohibitive in some parameter settings, we argue that for many regimes of interest, FLI+SOS’s concrete costs for proving accumulated instances are comparable to all alternatives. In particular, this is the case when $N^{1/c}$ is small, and m is relatively large. This scenario arises naturally in applications such as continuations in Jolt [1, 27] (cf. Sect. 1). All in all, we argue that FLI+SOS can be the best scheme “end-to-end”: for folding lookup instances, and proving the resulting accumulated instance.

Concretely, consider for example the setting $m = 2^{17}$, $N = 2^{1024}$, $N^{1/c} = 2^4$, $c = 256$, $\alpha = 2c$. Recall m is the size of the small table, N the size of the big table, which we assume is SOS-decomposable into α tables of size $N^{1/c}$ (see Definition 3). This setting resembles some of the parameter choices suggested in [1, 27]. Further, assume that we perform $n_f = 2^3$ folding steps, which would allow us to prove $m \cdot n_f = 2^{20}$ lookups. Based on these parameters, in Table 5 we use Tables 2 to 6 to approximate the costs of the folding Prover and the Prover for proving accumulated instances, counted in field multiplications and polynomial openings. Afterward, we briefly discuss how these costs were obtained, and mention possible variations in the schemes that would lead to other costs.

We discuss the setting where m is small and $N^{1/c}$ is relatively large in Appendix A. In such setting, FLI essentially only requires committing to binary vectors, due to the average case commitment cost of FLI (Tables 2 and 5).

Remark 2. (DT+SOS folding Prover cost) When it comes to DT+SOS, it is challenging to estimate the concrete costs of the folding Prover. We expect $\alpha \cdot \mathbf{L}$ to be the dominant cost, where recall \mathbf{L} denotes the cost of computing the coefficients of $e(X)$ in [7], which has degree 9 in this case. Even using FFT’s in the simplified scenario discussed in the footnote of Page 7 of [7], the cost would be over $\alpha \cdot 270m$ (Since computing $e(X)$ requires composing 9 homogeneous polynomials of degree $1, \dots, 9$ with a linear polynomial, m times). To this, one must add the cost α times the cost $(3m, [M])$ (committing to a $3m$ -sized vector with entries in $[M]$, where M is the largest entry in \mathbf{t}) plus the cost $\alpha \cdot P_{\text{SPS}}$ of

Table 5. Approximate costs of Protostar+SOS, DT+SOS, and FLI+SOS when $m = 2^{17}$, $N = 2^{1024}$, $N^{1/c} = 2^4$, and $\alpha = 2c = 256$. For each scheme, we display the cost of the folding Prover (in field multiplications) and of the Prover for accumulated instances (in field multiplications and group operations, in columns 3 and 4). Regarding the latter, we describe Provers for instances accumulated with Protostar+SOS and DT+SOS below (Sect. 6). In the second and last column, we assume that a MSM-based PCS is used over the Pallas curve, and use Table 1 in [17] to estimate the cost of these MSMs. *The costs of DT+SOS’s folding Prover are discussed in Remark 2 **These opening costs can be lowered, see the end of Sect. 6.

Scheme	Folding Prover	Acc. Prover	Openings	Rounds
Protostar+SOS	$\sim \alpha \cdot 2^{10.5} \cdot m$	$2^{13.32} \cdot m$	$\sim 2^{8.5} \cdot m$	$\log(m) + \alpha$
DT +SOS	*	$2^{13.25} \cdot m$	$\sim 2^{8.5} \cdot m$	$\log(m) + \alpha \cdot u \cdot \log(N^{1/c})$
FLI +SOS	$\left\{ \begin{array}{l} \text{avg.: } \sim \alpha \cdot 2^{8.6} \cdot \rho \\ \text{worse: } \sim \alpha \cdot 2^{8.5} \cdot m \end{array} \right.$ $\rho := \min\{mn_f/N^{1/c}, m\}$	$2^{13.59} \cdot m$	$\sim 2^{12} \cdot m^{**}$	$\log(m) + \alpha$

logUp-GKR’s Prover, and other costs. Because of this, we expect the final cost to be higher than FLI+SOS.

Provers for instances that were accumulated with Protostar+SOS or DT+SOS. The protocols that prove accumulated instances for Protostar and DT are unspecified in the original papers. Recall that in Sect. 1.2 we described variations of Protostar and DT that first perform the SOS decomposition step of Lasso, which reduces the initial claim into α lookup instances, and then folding each of the α instances with α accumulated instances, using Protostar or DT, respectively. Accordingly, one can imagine a scheme for Protostar or DT that proves each of the α accumulated lookup instances separately. In Protostar, the accumulated claim is roughly a statement that the identities used in logUp hold (cf. [5] or Lemma 5 in [17]). In DT, the same occurs, but this time the GKR variant of logUp is used [24]. For simplicity and for the sake of comparison, we assume that these claims can be proved, respectively, with logUp [17] and logUp-GKR [24]. We emphasize that, to our knowledge, no actual protocols for proving accumulated instances have been formally described, and that they may be more complex than the schemes we just sketched (typically error terms would appear in the logUp equations). With this in mind, in Table 6 we lower bound Protostar’s and DT’s costs for proving accumulated instances as the cost of running logUp (when using Protostar) or logUp-GKR (when using DT) on each of the resulting α accumulated claims that arise when applying the decomposition step of Lasso. We set the cost of logUp as $20 \max\{N^{1/c}, m\}$ field multiplications, and of logUp-GKR as $19 \max\{N^{1/c}, m\}$ field multiplications. We derive these costs for logUp and logUp-GKR using [11] in Appendix B.

Remark 3. In their simplest forms, logUp [17] and logUp-GKR [24] are lookups designed to handle instances where both the “small” and the “large” table have the same size. Both [17,24] have more elaborate versions that handle $m > N^{1/c}$ by splitting m into $m/N^{1/c}$ “columns” (following the terminology in [17,24]). When using this approach, as per [24], the Prover costs of Protostar and DT in Table 6 would incur a multiplicative overhead of $\approx \log(m/N^{1/c})$. Such a cost would make Protostar’s and DT’s costs for proving accumulated instances really high, and FLI would clearly be the best protocol. Because of this, and since it seems plausible, in our discussion we assume that there is a method that allows to treat the case $m > N^{1/c}$ by using one single column. We assume these improved protocols incur no overhead with respect to the original ones. The cost reflected in Table 6 corresponds to the dominant costs of such schemes. Further, the resulting cost for both Protostar and DT assume that we perform the sumchecks that prove logUp/logUp-GKR in parallel. In practice, this would lead to way too many evaluations, so the sumchecks would need to be batched. This results in an increase of concrete costs, though not significant.

Table 6. Dominant costs of the protocols for proving accumulated instances with Protostar+SOS, DT+SOS, and FLI+SOS. We follow the same notation as in Tables 1 to 3. The opening costs refer to computing opening proofs of multilinear polynomials, possibly dense. We assume polynomial evaluation claims are batched together using standard techniques, hence, for each number of variables, there is only one opening proof to be generated. For both Protostar and Deep Thought, the costs are approximate and based on a simplified scheme, cf. Remark 3

Scheme	Prover field mult	Openings
Protostar +SOS	$20\alpha \max\{m, N^{1/c}\}$	$1 \cdot \log(m)$ -variate, $1 \cdot \log(N^{1/c})$ -variate
Deep Thought + SOS	$19\alpha \max\{m, N^{1/c}\}$	$1 \cdot \log(m)$ -variate, $1 \cdot \log(N^{1/c})$ -variate
FLI + SOS	$2m(N^{1/c} + 1) + 3\alpha m \cdot n_f$	$1 \cdot \log(mN^{1/c})$ -variate

Prover for Accumulated Instances’ Openings Costs. FLI’s scheme for proving accumulated instances requires proving an evaluation of a $\log(mN^{1/c})$ -variate (sparse) multilinear polynomial. We assume that a curve-based commitment scheme is used, and that the opening proof bottleneck occurs when computing Multi-Scalar-Multiplication (MSM) of size $mN^{1/c}$. We remark that due to #2 in [27], certain variations of Zeromorph [19] or HyperKZG [25] might make this step not be a bottleneck anymore. The price to pay is an increase of the folding Verifier work by $O(N^{1/c})$ (which can be configured to be small). Indeed, this is the approach planned by the Jolt team [27]. In other words, here we analyze the costs of a naive selection of commitment scheme, but we remark that there are plausible alternatives.

Following the benchmarks in Table 1 of [17], we set the cost of this MSM when $mN^{1/c} > 2^{18}$ to be around $2^8 mN^{1/c}$ field multiplications, on the Pallas curve. With our parameter choice, this is roughly $2\times$ less expensive than Protostar and DT’s cost for proving accumulated instances, which is $\approx 2^{13.2}m$.

A Comparing Other Regimes for m and N

The case where m is small and $N^{1/c}$ is large. Assuming $N^{1/c}$ is large and both m and n_f are small, then in general FLI’s commitment cost is very small (on average), coming from committing to c vectors of, mostly, binary elements. In this scenario, we can reasonably assume that FLI’s folding cost is dominated by $m \deg(g)(\alpha + |g|)$ field operations. As shown in Tables 2 and 3, in this parameter regime, FLI results in the most efficient folding scheme available, for many parameter choices. Further, towards building an IVC/PCD scheme, FLI’s Verifier is the cheapest among the schemes in Tables 2 and 3.

If n_f is large, then ρ may degenerate to m , in which case FLI’s folding Prover must commit to c size- m vectors of arbitrarily sized field elements. Still, Protostar must commit to $2\alpha = 2 \cdot k \cdot c$ size- m such vectors. This is roughly $2k$ times more expensive than FLI’s commitment costs. Further, we note that the overall number of commitments to large elements over the n_f folding steps is, in FLI, on average:

$$c \frac{m}{N^{1/c}} (1 + 2 + \dots + n_f) \approx \frac{cn_f^2 m}{2N^{1/c}},$$

while in Protostar it is $2\alpha n_f m$.

The case when both m and $N^{1/c}$ are large. So far, we have only discussed the case when $mN^{1/c}$ is roughly our computation target. This limited our analysis to settings where m and/or $N^{1/c}$ were “small”. The reason for this analysis is that FLI’s protocol for proving accumulated instances has cost $O(mN^{1/c})$, and thus cannot handle scenarios where both m and $N^{1/c}$ are large. On the other hand, Protostar and Deep Thought do not present the quadratic term $mN^{1/c}$, and so it is only fair to remark that one may use them in this regime, while FLI is not usable in this case.

We note however that, when dealing with SOS-decomposable tables as the ones from Jolt, it is always possible to select c so that $N^{1/c}$ is small. Hence, when looking into such tables, there is always the option of selecting parameters as in Sect. 6 and use FLI.

B The Cost of Proving Accumulated Instances with Protostar+SOS and DT+SOS

As discussed in Sect. 6, we use an oversimplification and estimate the cost of proving an accumulated instance with Protostar+SOS (resp. DT+SOS) by estimating the cost of proving α sumchecks appearing in $\log\text{Up}$ (resp. $\log\text{Up-GKR}$) when applied to a lookup of a table of size $\max\{m, N^{1/c}\}$ into a table of the same size. In turn, we estimate the cost of proving a single such sumcheck using the new optimized costs in [11].

The cost of logUp [17] For logUp, the sumcheck for a lookup of a small table \mathbf{a} into a big table \mathbf{t} (both of size $M := \max\{m, N^{1/c}\}$) is of the form:

$$\sum_{\mathbf{x} \in \mathbb{B}^{\log(M)}} h_1(\mathbf{x}) + h_2(\mathbf{x}) + \gamma \cdot \tilde{\mathbf{e}}\mathbf{q}(\mathbf{x}; \boldsymbol{\beta}) \cdot (h_1(\mathbf{x}) \cdot (\alpha + \mathbf{t}(\mathbf{x})) - \mathbf{m}(\mathbf{x})) + \tag{*}$$

$$\gamma^2 \cdot \tilde{\mathbf{e}}\mathbf{q}(\mathbf{x}; \boldsymbol{\beta}) \cdot (h_2(\mathbf{x}) \cdot (\alpha + \mathbf{a}(\mathbf{x})) - 1) = 0$$

where $\alpha \in \mathbb{F}$ and $\boldsymbol{\beta} \in \mathbb{F}^{\log(M)}$ are chosen at random, \mathbf{m} is the "multiplicity function" (c.f. [17], this functions indicates how many times each entry $\mathbf{t}(\mathbf{x})$ is looked up) and:

$$\forall \mathbf{x} \in \mathbb{B}^{\log(M)}, h_1(\mathbf{x}) := \mathbf{m}(\mathbf{x}) / (\alpha + \mathbf{t}(\mathbf{x}))$$

$$h_2(\mathbf{x}) := -1 / (\alpha + \mathbf{a}(\mathbf{x}))$$

To apply the sumcheck optimizations of [11], we need all the evaluations of h_1, h_2 . Batch inversion techniques allow to compute the vector of inverses of $(\alpha + \mathbf{t}(\mathbf{x}), \alpha + \mathbf{a}(\mathbf{x}))_{\mathbf{x}}$ in about $6M$ multiplications, and 1 inversion. We can then use M multiplications to compute all evaluations of h_1, h_2 over the hypercube. Following [11], the Prover in sumcheck in (*) performs:

- $2M$ field multiplications for the terms $h_1(\mathbf{x})$ and $h_2(\mathbf{x})$.
- $6M + O(\sqrt{M})$ field multiplications for the term $\gamma \cdot \tilde{\mathbf{e}}\mathbf{q}(\mathbf{x}; \boldsymbol{\beta}) \cdot (h_1(\mathbf{x}) \cdot (\alpha + \mathbf{t}(\mathbf{x})) - \mathbf{m}(\mathbf{x}))$.
- $6M$ field multiplications for the term $\gamma^2 \cdot \tilde{\mathbf{e}}\mathbf{q}(\mathbf{x}; \boldsymbol{\beta}) \cdot (h_2(\mathbf{x}) \cdot (\alpha + \mathbf{a}(\mathbf{x})) - 1)$.

For a total cost of $20M + O(\sqrt{M})$ field multiplications.

The cost of logUp-GKR [24]. For logUp-GKR, for a lookup of a small table \mathbf{a} into a big table \mathbf{t} (both of size $M := \max\{m, N^{1/c}\}$), there are k sumchecks (for $1 \leq k \leq \log(M) - 1$) of the form:

$$\sum_{\mathbf{x} \in \mathbb{B}^k} \tilde{\mathbf{e}}\mathbf{q}(\mathbf{x}; \boldsymbol{\beta}) \cdot (p_{k+1}(\mathbf{x}, 1) \cdot q_{k+1}(\mathbf{x}, -1) + p_{k+1}(\mathbf{x}, -1) \cdot q_{k+1}(\mathbf{x}, 1) +$$

$$\gamma_k \cdot q_{k+1}(\mathbf{x}, 1) \cdot q_{k+1}(\mathbf{x}, -1)) = p_k(\boldsymbol{\beta}) + \gamma_k \cdot q_k(\boldsymbol{\beta}) \tag{†}$$

where $\alpha \in \mathbb{F}$ and $\boldsymbol{\beta} \in \mathbb{F}^k$ are chosen at random, and:

$$p(\mathbf{X}, Y) := Y \cdot \mathbf{m}(\mathbf{X}) - (1 - Y)$$

$$q(\mathbf{X}, Y) := Y \cdot (\alpha - \mathbf{t}(\mathbf{X})) + (1 - Y) \cdot (\alpha - \mathbf{a}(\mathbf{X}))$$

$$\forall 1 \leq k \leq \log(M) - 1, \forall \mathbf{x} \in \mathbb{B}^k, \frac{p_k(\mathbf{x})}{q_k(\mathbf{x})} = \sum_{\mathbf{y} \in \mathbb{B}^{\log(M)-k}} \frac{p(\mathbf{x}, \mathbf{y})}{q(\mathbf{x}, \mathbf{y})}$$

At layer k , following [11] and assuming that we have all relevant values of p_{k+1}, q_{k+1} , the Prover can perform the sumcheck in (†) in $16 \cdot 2^k + O(2^{k/2})$ field

multiplications. Summing over k , this leads to $16M + O(\sqrt{M})$ field multiplications overall. To compute all necessary values of the p_k, q_k for all k , we need $3M$ multiplications (this is because of the linear relationships between p_k, q_k and p_{k+1}, q_{k+1} , see [24]). All in all, we estimate that the Prover in logUp-GKR performs no more than $19M + O(\sqrt{M})$ field multiplications.

References




1. Arun, A., Setty, S., Thaler, J.: Jolt: Snarks for virtual machines via lookups. In: Joye, M., Leander, G. (eds.) *Advances in Cryptology – EUROCRYPT 2024*. pp. 3–33. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-58751-1_1
2. Arun, A., Zhu, M.: Jolt: Snarks for virtual machines via lookups. *ZK Proof Standards (2024)*, <https://www.youtube.com/live/RySXjCsLgXk>
3. barryWhiteHat: Lookup singularity. *ZKResearch (2022)*, <https://zkresearch.ch/t/lookup-singularity/65>
4. Bowe, S., Grigg, J., Hopwood, D.: Recursive proof composition without a trusted setup. *Cryptology ePrint Archive, Paper 2019/1021 (2019)*, <https://eprint.iacr.org/2019/1021>
5. Bünz, B., Chen, B.: Protostar: Generic efficient accumulation/folding for special-sound protocols. In: *Advances in Cryptology - ASIACRYPT 2023: 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part II*. p. 77–110. Springer-Verlag, Berlin, Heidelberg (2023). https://doi.org/10.1007/978-981-99-8724-5_3
6. Bünz, B., Chiesa, A., Lin, W., Mishra, P., Spooner, N.: Proof-carrying data without succinct arguments. In: Malkin, T., Peikert, C. (eds.) *Advances in Cryptology – CRYPTO 2021*. pp. 681–710. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-84242-0_24
7. Bünz, B., Chen, J.: Proofs for deep thought: Accumulation for large memories and deterministic computations. *Cryptology ePrint Archive, Paper 2024/325 (2024)*, <https://eprint.iacr.org/2024/325>
8. Bünz, B., Chiesa, A., Mishra, P., Spooner, N.: Proof-carrying data from accumulation schemes. *Cryptology ePrint Archive, Paper 2020/499 (2020)*, <https://eprint.iacr.org/2020/499>
9. Campanelli, M., Gailly, N., Gennaro, R., Jovanovic, P., Mihali, M., Thaler, J.: Testudo: Linear time prover snarks with constant size proofs and square root size universal setup. In: Aly, A., Tibouchi, M. (eds.) *Progress in Cryptology – LATINCRYPT 2023*. pp. 331–351. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-44469-2_17
10. Chiesa, A., Tromer, E.: Proof-carrying data and hearsay arguments from signature cards. In: *ICS*. pp. 310–331. Tsinghua University Press (2010)
11. Dao, Q., Thaler, J.: More optimizations to sum-check proving. *Cryptology ePrint Archive, Paper 2024/1210 (2024)*, <https://eprint.iacr.org/2024/1210>
12. Diamond, B.E., Posen, J.: Succinct arguments over towers of binary fields. *Cryptology ePrint Archive, Paper 2023/1784 (2023)*, <https://eprint.iacr.org/2023/1784>
13. Diamond, B.E., Posen, J.: Proximity testing with logarithmic randomness. *IACR Communications in Cryptology* **1**(1) (2024). <https://doi.org/10.62056/aksdkp10>

14. Eagen, L., Fiore, D., Gabizon, A.: cq: Cached quotients for fast lookups. *Cryptology ePrint Archive*, Paper 2022/1763 (2022), <https://eprint.iacr.org/2022/1763>
15. Eagen, L., Gabizon, A.: Protogalaxy: Efficient protostar-style folding of multiple instances. *Cryptology ePrint Archive*, Paper 2023/1106 (2023), <https://eprint.iacr.org/2023/1106>
16. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: *Advances in Cryptology - CRYPTO 1986*, Santa Barbara, California, USA, 1986, Proceedings. *Lecture Notes in Computer Science*, vol. 263, pp. 186–194. Springer (1986). https://doi.org/10.1007/3-540-47721-7_12
17. Haböck, U.: Multivariate lookups based on logarithmic derivatives. *Cryptology ePrint Archive*, Paper 2022/1530 (2022), <https://eprint.iacr.org/2022/1530>
18. Kilian, J.: A note on efficient zero-knowledge proofs and arguments (extended abstract). In: *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, p. 723–732. STOC '92, Association for Computing Machinery, New York, NY, USA (1992). <https://doi.org/10.1145/129712.129782>
19. Kohrita, T., Towa, P.: Zeromorph: Zero-knowledge multilinear-evaluation proofs from homomorphic univariate commitments. *Cryptology ePrint Archive*, Paper 2023/917 (2023), <https://eprint.iacr.org/2023/917>
20. Kothapalli, A., Setty, S.: Hypernova: Recursive arguments for customizable constraint systems. In: Reyzin, L., Stebila, D. (eds.) *Advances in Cryptology – CRYPTO 2024*. pp. 345–379. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-68403-6_11
21. Kothapalli, A., Setty, S., Tzialla, I.: Nova: Recursive zero-knowledge arguments from folding schemes. In: Dodis, Y., Shrimpton, T. (eds.) *Advances in Cryptology – CRYPTO 2022*. pp. 359–388. Springer Nature Switzerland, Cham (2022). https://doi.org/10.1007/978-3-031-15985-5_13
22. Micali, S.: Cs proofs. In: *Proceedings 35th Annual Symposium on Foundations of Computer Science*. pp. 436–453 (1994). <https://doi.org/10.1109/SFCS.1994.365746>
23. Papamanthou, C., Shi, E., Tamassia, R.: Signatures of correct computation. In: Sahai, A. (ed.) *Theory of Cryptography*. pp. 222–242. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36594-2_13
24. Papini, S., Haböck, U.: Improving logarithmic derivative lookups using gkr. *Cryptology ePrint Archive*, Paper 2023/1284 (2023), <https://eprint.iacr.org/2023/1284>
25. Setty, S.: Hyperkzg (2024), <https://github.com/microsoft/Nova/blob/main/src/provider/hyperkzg.rs>
26. Setty, S., Thaler, J., Wahby, R.: Unlocking the lookup singularity with lasso. In: Joye, M., Leander, G. (eds.) *Advances in Cryptology – EUROCRYPT 2024*. pp. 180–209. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-58751-1_7
27. Thaler, J.: *Faq on jolt's initial implementation* (2024), <https://a16zcrypto.com/posts/article/faqs-on-jolts-initial-implementation/>
28. Valiant, P.: Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In: Canetti, R. (ed.) *Theory of Cryptography*. pp. 1–18. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78524-8_1

29. Zapico, A., Buterin, V., Khovratovich, D., Maller, M., Nitulescu, A., Simkin, M.: Caulk: Lookup arguments in sublinear time. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. p. 3121-3134. CCS '22, Association for Computing Machinery, New York, NY, USA (2022).<https://doi.org/10.1145/3548606.3560646>
30. Zapico, A., Gabizon, A., Khovratovich, D., Maller, M., Ràfols, C.: Baloo: Nearly optimal lookup arguments. Cryptology ePrint Archive, Paper 2022/1565 (2022), <https://eprint.iacr.org/2022/1565>



Code-Based Zero-Knowledge from VOLE-in-the-Head and Their Applications: Simpler, Faster, and Smaller

Ying Ouyang , Deng Tang , and Yanhong Xu 

Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China
{dengtang, yanhong.xu}@sjtu.edu.

Abstract. Zero-Knowledge (ZK) protocols allow a prover to demonstrate the truth of a statement without disclosing additional information about the underlying witness. Code-based cryptography has a long history but did suffer from periods of slow development. Recently, a prominent line of research have been contributing to designing efficient code-based ZK from MPC-in-the-head (Ishai et al., STOC 2007) and VOLE-in-the-head (VOLEitH) (Baum et al., Crypto 2023) paradigms, resulting in quite efficient standard signatures. However, none of them could be directly used to construct privacy-preserving cryptographic primitives. Therefore, Stern’s protocols remain to be the major technical stepping stones for developing advanced code-based privacy-preserving systems.

This work proposes new code-based ZK protocols from VOLEitH paradigm for various relations and designs several code-based privacy-preserving systems that considerably advance the state-of-the-art in code-based cryptography. Our first contribution is a new ZK protocol for proving the correctness of a regular (non-linear) encoding process, which is utilized in many advanced privacy-preserving systems. Our second contribution are new ZK protocols for concrete code-based relations. In particular, we provide a ZK of accumulated values with optimal witness size for the accumulator (Nguyen et al., Asiacrypt 2019). Our protocols thus open the door for constructing more efficient privacy-preserving systems. Moreover, our ZK protocols have the advantage of being simpler, faster, and smaller compared to Stern-like protocols. To illustrate the effectiveness of our new ZK protocols, we develop ring signature scheme, group signature scheme, fully dynamic attribute-based signature scheme from our new ZK. The signature sizes of the resulting schemes are two to three orders of magnitude smaller than those based on Stern-like protocols in various parameter settings. Finally, our first ZK protocol yields a standard signature scheme, achieving “signature size + public key size” as small as 3.05 KB, which is slightly smaller than the state-of-the-art signature scheme (Cui et al., PKC 2024) based on the regular syndrome decoding problems.

Keywords: Zero-knowledge protocols · VOLE-in-the-head · code-based cryptography · privacy-preserving schemes · signature scheme

1 Introduction

A beautiful and fundamental notion introduced by Goldwasser, Micali and Rackoff [46], zero-knowledge (ZK) proof allows to prove a statement while not revealing anything about the witness. In the last three decades or so, ZK protocols are an important tool in designing numerous cryptographic constructions. Thanks to the Fiat-Shamir heuristic [44], ZK protocols have been the basis for developing standard signatures and privacy-enhancing authentication systems, such as group signature (GS) [31], ring signature (RS) [73], attribute-based signatures (ABS) [25], anonymous credential (AC) [30], and policy-based signature (PBS) [12].

Traditional cryptographic schemes based on number-theoretic assumptions are at the risk of being broken by quantum computers. This threat motivates the research for new ZK proof techniques based on post-quantum cryptographic problems. Among all possible alternatives, code-based cryptography is one of the promising choices. Dating back to 1996, Stern [76] introduced the first ZK for syndrome decoding (SD) problem and the framework has been utilized for constructing code-based signatures and privacy-preserving systems.

However, Stern protocols and its followup works [39, 56, 57] have soundness error $2/3$, preventing it from being practical. Therefore, numerous works (e.g., [19, 50, 61, 83]) have been devoting to construct more efficient protocols with smaller soundness error. A recent line of research in code-based cryptography by Gueron et al. [47], Bidoux et al. [19] and Feneuil et al. [41], have independently lowered the soundness error to $1/N$ for an arbitrary N by leveraging a technique inspired from the well-known MPC-in-the-head (MPCitH) paradigm [49, 51]. Since then, MPCitH and its recent variant VOLE-in-the-head (VOLEitH) [8] have achieved a high success in designing efficient code-based ZK proofs and standard signature schemes [1, 18, 26, 29, 32, 40, 42, 64, 65].

To the best of our knowledge, none of these ZK protocols could be directly used to construct advanced privacy-preserving primitives from codes, where more sophisticated algebraic structures are required. In particular, a prominent line of research in designing code-based privacy-preserving schemes employed accumulators [14] to achieve logarithmic proof sizes [56, 68, 69, 79]. The main technical difficulty of utilizing accumulators in designing these schemes is a supporting ZK argument of valid accumulated values. This is particularly challenging for the code-based accumulators [69] built from Merkle hash trees [66]. This is because the output of each hashing has to be encoded to a small-weight vector (with respect to its dimension) before going to the next step and we have to prove that the whole recursive process is done correctly. To overcome this difficulty, Nguyen et al. [69] designed a dedicated and involved (thus inefficient) ZK protocol to prove the correctness of the encoding process within Stern's framework. We note that a recent work by Ling et al. [56] has revisited the long-established Stern's protocol and put forward a new refined framework. Theoretically interesting and beautiful, the refined framework has not yielded noteworthy efficiency improvement. Nevertheless, Stern-like protocols remain to be the major technical

stepping stone for developing code-based advanced privacy-preserving systems, even they are still far from being practical.

In this work, we aim to contribute to the development of practically efficient ZK protocols for codes, particularly for proving the knowledge of accumulated values, which can be further used to construct various advanced privacy-preserving primitives. Since all the ZK protocols presented in this work belong to the VOLEitH paradigm [8]. Let us briefly review the development of it.

Vector oblivious linear evaluation (VOLE)-based ZK protocols were initiated by Boyle [21, 23]. Due to low memory consumption and linear (in the circuit) proof sizes, VOLE-based ZK protocols have recently seen a lot of progress [9–11, 22, 34, 35, 55, 80–82]. At a high level, VOLE-based proofs employ preprocessed random VOLE correlations to implement highly efficient proofs via a commit-and-prove paradigm. Recently, Baum et al. [8] developed a new method, named VOLEitH, resulting in simpler, faster, and smaller proofs than related approaches based on MPCitH [49]. They then instantiated their paradigm with two protocols, one for proving statements over large fields, and the other for proving statements over small fields. In addition, they briefly mentioned how to extend their protocols to proving low-degree polynomials satisfiability via the techniques from the QuickSilver [82] protocol.

Due to the attractive features of VOLEitH paradigm, Cui et al. [32] designed a new ZK for proving the knowledge of a solution to the regular syndrome decoding (RSD) problem using this paradigm, and turned their ZK into a standard signature scheme ReSolveD. Bidoux et al. [18] also applied VOLEitH to prove solutions to rank SD and MinRank problems, and obtained efficient code-based signatures.

1.1 Our Contributions

In this work, we provide a brand new ZK protocol for proving the correctness of a regular encoding process within the VOLEitH paradigm. Built upon this core technique, we then provide efficient ZK arguments of knowledge of valid opening, of an accumulated value, and of a plaintext. As main applications of our ZK protocols, we construct efficient RS, GS, fully dynamic ABS (FDABS) schemes whose signature sizes are two to three orders of magnitude smaller than those based on Stern-like ZK protocols. In addition, our new ZK protocols naturally yield a standard signature scheme, which is as efficient as the state-of-the-art code-based ones [18, 32] with a flexible tradeoff on communication and computation.

Contribution to ZK Protocol for Proving the Correctness of a Regular Encoding Process. Recall that Nguyen et al. [69] employed the following regular encoding function to build their accumulator. Let c be a positive integer. Given a binary vector $\mathbf{x} = (x_1, \dots, x_c)^\top$, let $t = \sum_{h=1}^c 2^{c-h} \cdot x^h$ be the integer whose binary representation is exactly \mathbf{x} . $\text{RE} : \{0, 1\}^c \rightarrow \{0, 1\}^{2^c}$ maps \mathbf{x} to $\mathbf{y} = \text{RE}(\mathbf{x})$, where \mathbf{y} is the unit vector of length 2^c with the sole 1 at the $(t+1)$ -th position. To demonstrate that \mathbf{y} is a correct regular encoding of \mathbf{x} , Nguyen et

al. [69] employed a dedicated permutation technique that works well in Stern’s framework. However, this permutation technique prohibits the statement about the correct regular encoding process from being proved in other more efficient MPCitH or VOLEitH frameworks. To improve the efficiency, we instead take one step back and observe that it suffices to express the regular encoding process into polynomial constraints, a set of statements that can be proved within the VOLEitH framework. To this end, we reinterpret the regular encoding process as 2^c Boolean functions, which can be seen as a special case of polynomial constraints. In addition, these Boolean functions have degree c , which is usually a small constant ranging from 2 to 8. Therefore, our targeted statement can be *efficiently* proved within the VOLEitH paradigm.

We remark that this regular encoding function is employed in designing code-based commitment schemes and accumulators [69], which are essential building blocks for many privacy-preserving schemes, e.g., [56, 68, 69, 79]. Therefore, our new ZK protocols open the door for constructing more efficient code-based privacy-preserving schemes such as RS, GS, ABS, AC, PBS.

Contribution to ZK Protocols for Concrete Code-based Relations.

Building upon the core technique of proving the correct encoding process, we propose a variety of ZK for some concrete code-based relations that are essential in constructing privacy-enhancing authentication systems. In particular, we provide a new ZK protocol for proving the knowledge of committed values for the commitment scheme [69], a ZK protocol for proving the knowledge of accumulated values for the accumulator [69], and a ZK protocol for proving the knowledge of plaintexts for a variant of McEliece cryptosystem [63, 70].

All our ZK protocols are within VOLEitH paradigm. In more detail, we reduce the above tasks to proving polynomial constraints through careful transformation. Importantly, proving the correctness of the regular encoding process $\mathbf{y} = \text{RE}(\mathbf{x})$ essentially implies that \mathbf{y} is a regular word. This observation is a key to huge efficiency improvement. Let us elaborate it more. When proving the knowledge of an accumulated value, \mathcal{P} is to prove the knowledge of $(j_1, \dots, j_\ell)^\top \in \{0, 1\}^n$, $\mathbf{v}_1, \mathbf{w}_1, \dots, \mathbf{v}_\ell, \mathbf{w}_\ell \in \mathbb{F}_2^n$ such that

$$\forall i \in \{\ell - 1, \dots, 1, 0\}, \mathbf{v}_i = \begin{cases} \mathbf{B}_0 \cdot \text{RE}(\mathbf{v}_{i+1}) + \mathbf{B}_1 \cdot \text{RE}(\mathbf{w}_{i+1}), & \text{if } j_{i+1} = 0; \\ \mathbf{B}_0 \cdot \text{RE}(\mathbf{w}_{i+1}) + \mathbf{B}_1 \cdot \text{RE}(\mathbf{v}_{i+1}), & \text{if } j_{i+1} = 1. \end{cases} \quad (1)$$

Here $\text{RE} : \{0, 1\}^n \rightarrow \{0, 1\}^{\frac{n}{c} \cdot 2^c}$. As mentioned earlier, \mathcal{P} has to prove that the above recursive steps are done correctly. Particularly, \mathcal{P} has to demonstrate that $\text{RE}(\mathbf{v}_i)$ is indeed a regular encoding of \mathbf{v}_i . This can be proved via Stern’s protocol [69, 76] or our ZK. However, Nguyen et al. [69] had to introduce intermediate vectors $\mathbf{v}'_{i+1} = \text{RE}(\mathbf{v}_{i+1})$ and $\mathbf{w}'_{i+1} = \text{RE}(\mathbf{w}_{i+1})$, thus blowing up the witness size from $\ell + 2\ell n$ bits to $(2\ell) \cdot \frac{2n}{c} \cdot 2^c + 2(\ell - 1) \cdot n$ bits. This blowup also results from the involved methods they developed to remove the dependence on j_1, \dots, j_ℓ when computing \mathbf{v}_0 . Our protocol, in contrast, can achieve the optimal witness size $\ell + 2\ell n$. As a result, our new ZK protocols have the advantage of being simpler, faster, and smaller compared to Stern-like protocols.

We emphasize that our ZK of accumulated values for accumulators built from Merkle trees is the first one that achieves optimal witness size $\ell + 2\ell n$. This is one of the main reasons that our new ZK protocols outperform Stern-like protocols.

Contribution to Code-Based Advanced Privacy-preserving Primitives.

To further illustrate the effectiveness of our new techniques, we develop several code-based privacy-preserving primitives from these new ZK protocols. In particular, we provide new ZK protocols for the ring signature scheme [69], the group signature scheme [69], the fully dynamic attribute-based signature scheme [56]. In addition, we examine the concrete signature sizes of our ZK protocols, and compare the results with the (refined) Stern-like ZK. Details are given in Table 3, Table 4, and Table 5. The comparisons exhibit the superiority of our new ZK protocols, which are two to three orders of magnitude smaller than Stern-like protocols in various parameter settings.

Table 1. Comparison of signature sizes for different privacy-preserving primitives from different ZK protocols based on various post-quantum assumptions for 128-bit security and ring/group size 2^{10} . Note that Katz et al. [51] evaluated the signature sizes at 256-bit security. We then include the sizes of our ZK at the same level below their results. For FDABS, the maximum number of attributes is $2^\ell = 2^{10}$, and the size of the circuit P is $K = 2^9$.

Schemes	code-based This work	code-based (Stern-type)		hash-based [51]	lattice-based [62]
RS	60 KB	61 MB [69]	61 KB [60]	388 KB (240 KB)	13 KB
GS	75 KB	63 MB [69]	121 KB* [60]	418 KB** (297 KB)	18 KB*
FDABS	62 KB	46 MB [56]	-	-	-

*: They only achieve CPA-anonymity.

** : It only achieves selfless anonymity.

Next, we give a brief comparison between the signature sizes of privacy-preserving schemes in this work and those of some previous post-quantum constructions. The results¹ are summarized in Table 1, in which we target for 128-bit security and ring/group size 2^{10} . The comparison shows that our ZK protocols perform much better, around three orders of magnitude smaller, than Stern-like ZK in [56, 69]. Compared to the code-based ring signature and group signature schemes by Liu and Wang [60], our ring signature are as efficient as theirs while our group signature sizes are around 40% smaller. Also, our ring/group signature sizes are around 30%~40% smaller than the state-of-the-art hash-based

¹ El Kaafarani and Katsumata [36] presented a lattice-based ABS scheme without giving concrete efficiency analysis. Since they employed Stern's protocols, their ABS scheme is supposed to be less efficient than [56].

ones by Katz et al. [51]. Moreover, our performances are comparable to the state-of-the-art lattice-based constructions [62], in which the signature sizes are 13 KB (for RS) and 18 KB (for GS) in a similar parameter setting. We stress that they [62] employed the nice features of structured lattices and specialized techniques for optimal efficiency, and the three GS constructions [51, 60, 62] only achieve weaker form of anonymity. In contrast, our new ZK protocols are able to design CCA-anonymous GS and more advanced primitives such as FDABS.

We remark that the applications of our ZK protocols to RS, GS, FDABS are by no means exhaustive nor optimal. In fact, it is possible to employ our ZK protocols to design more efficient code-based privacy-preserving schemes such as group encryption [52], AC, PBS. Also, one can improve the performance of our ZK by choosing less conservative parameters for the underlying accumulator [69] or smaller parameters for the McEliece encryption scheme. We leave those extensions and optimizations to future work.

A New Signature Scheme Based on RSD Problem. Finally, we give a new signature scheme ReSolveD+ (improving upon ReSolveD [32]) based on the hardness of regular syndrome decoding (RSD) problem [4, 5]. The construction follows from the crucial observation that \mathbf{y} is a regular word if \mathbf{y} is a correct regular encoding of some secret vector \mathbf{x} , and from the standard methodology of turning a public-coin ZK protocol into a signature scheme via the Fiat-Shamir heuristic [44]. We provide various parameter sets that offer tradeoffs between communication and computation targeting 128-bit security. The shortest version of our signature scheme achieves “signature size + public key size” 3.05 KB, which is slightly smaller than the state-of-the-art code-based signature schemes [18, 32] based on RSD and a less studied rank SD problem. We give a detailed comparison of our signature scheme with previous works in Table 8.

1.2 Technical Overview

Let us now give a high-level discussion for our contributions.

ZK for Regular Encoding Process. Recall that we need to represent the regular encoding process $\text{RE} : \{0, 1\}^c \rightarrow \{0, 1\}^{2^c}$ into polynomial constraints. Towards this goal, we observe that RE can be seen as 2^c Boolean functions $f_{(0, \dots, 0)}(X_1, \dots, X_c), f_{(0, \dots, 0, 1)}(X_1, \dots, X_c), \dots, f_{(1, \dots, 1)}(X_1, \dots, X_c)$. So the next question is whether we could explicitly give out these Boolean functions. The answer turns out to be affirmative. Through simple yet non-trivial calculation, the truth table of $f_{(j_1, \dots, j_c)}(X_1, \dots, X_c)$ is exactly the unit vector \mathbf{e}_j , where $(j_1, \dots, j_c)^\top$ is the binary representation of $(j - 1)$. Then by Lagrange interpolation, we can explicitly express $f_{(j_1, \dots, j_c)}(X_1, \dots, X_c) = \prod_{h=1}^c (1 + j_h + X_h)$. At this point, we have successfully transformed the regular encoding process into degree- c relations and the witness size is exactly c bits.

ZK of a Valid Opening. We now describe how to construct a new and more efficient ZK argument of knowledge of a valid opening for the commitment

scheme [69]. The prover is to prove the knowledge of $\mathbf{x} \in \mathbb{F}_2^L, \mathbf{r} \in \mathbb{F}_2^k$ such that

$$\mathbf{c} = \mathbf{B}_0 \cdot \text{RE}(\mathbf{x}) + \mathbf{B}_1 \cdot \text{RE}(\mathbf{r}) \in \mathbb{F}_2^n, \quad (2)$$

with $\mathbf{B}_0 \in \mathbb{F}_2^{n \times \frac{L}{c} \cdot 2^c}$ and $\mathbf{B}_1 \in \mathbb{F}_2^{n \times \frac{k}{c} \cdot 2^c}$. As we are able to represent $\text{RE}(\mathbf{x})$ and $\text{RE}(\mathbf{r})$ as $(f_1(\mathbf{x}), \dots, f_{\frac{L}{c} \cdot 2^c}(\mathbf{x}))^\top$ and $(f_{\frac{L}{c} \cdot 2^c + 1}(\mathbf{r}), \dots, f_{\frac{L+k}{c} \cdot 2^c}(\mathbf{r}))^\top$, Eq. (2) can be easily transformed to n polynomials that are linear combinations of f_i for $i \in [1, \frac{L+k}{c} \cdot 2^c]$ subtracted by constants.

We remark that it is possible to employ the same linear sketching techniques as in [32] to show that $\text{RE}(\mathbf{x})$ and $\text{RE}(\mathbf{r})$ are regular words. However, this would incur witness size $\frac{L+k}{c} \cdot 2^{c^2}$ instead of the optimal witness size $L + k$ achieved by using our techniques.

ZK of an Accumulated Value. Recall that the goal of \mathcal{P} is to prove knowledge of $(j_1, \dots, j_\ell)^\top \in \{0, 1\}^n, \mathbf{v}_1, \mathbf{w}_1, \dots, \mathbf{v}_\ell, \mathbf{w}_\ell \in \mathbb{F}_2^n$ such that (1) hold. This task can be divided into three parts: (i) demonstrate that $\text{RE}(\mathbf{v}_1), \dots, \text{RE}(\mathbf{v}_\ell), \text{RE}(\mathbf{w}_1), \dots, \text{RE}(\mathbf{w}_\ell)$ are regular words; (ii) demonstrate that the branches of the tree is correctly chosen according to j_1, \dots, j_ℓ ; (iii) demonstrate that $\text{RE}(\mathbf{v}_i)$ is a correct regular encoding of \mathbf{v}_i for $i \in [1, \ell]$. We have seen that (i) can be proved via our techniques or the linear sketching techniques [32]. However, the latter would deteriorate the efficiency. In particular, the linear sketching techniques would incur witness size $2\ell \cdot \frac{n}{c} \cdot 2^c$ while our techniques only incur witness size $2\ell \cdot n$. We thus stick to our techniques. Regarding (ii), let $\overline{j_{i+1}} = 1 - j_{i+1}$, then we observe that (1) is equivalent to

$$\mathbf{v}_i = \mathbf{B}_0 \cdot (\overline{j_{i+1}} \text{RE}(\mathbf{v}_{i+1}) + j_{i+1} \text{RE}(\mathbf{w}_{i+1})) + \mathbf{B}_1 \cdot (\overline{j_{i+1}} \text{RE}(\mathbf{w}_{i+1}) + j_{i+1} \text{RE}(\mathbf{v}_{i+1})),$$

where $\mathbf{B}_0, \mathbf{B}_1 \in \mathbb{F}_2^{n \times \frac{n}{c} \cdot 2^c}$. Thus, the terms $j_{i+1} \cdot \text{RE}(\mathbf{v}_{i+1})$ and $j_{i+1} \cdot \text{RE}(\mathbf{w}_{i+1})$ can be represented as $(f'_1(\cdot), \dots, f'_{\frac{n}{c} \cdot 2^c}(\cdot))^\top$ and $(f'_{\frac{n}{c} \cdot 2^c + 1}(\cdot), \dots, f'_{2 \cdot \frac{n}{c} \cdot 2^c}(\cdot))^\top$, in which the degree of each polynomial f'_i increases to $(c+1)$ due to multiplication with the secret bit j_{i+1} . Similar to the above ZK of a valid opening, equations in (1) can now be transformed to ℓn polynomials. One then observes that (iii) is naturally solved if using our ZK for proving (i). We remark that the linear sketching techniques [32] cannot be used to prove (iii). In fact, they mainly focused on proving knowledge of a regular word and did not involving any regular encoding process, let alone prove correct regular encoding process.

We would also like to stress that the above simplicity for proving (ii) and (iii) only benefits from the fact that we represent the regular encoding process as polynomials and work in the VOLEitH paradigm. In fact, in a similar setting of proving the knowledge of an accumulated value, Libert et al. [54], Nguyen et al. [69], Yang et al. [83], Derler et al. [33], Boneh et al. [20] developed quite sophisticated and dedicated techniques to prove the honest computation of \mathbf{v}_i

² They introduced an optimization that can reduce the witness size to $\frac{L+k}{c} \cdot 2^c - n$, which is still larger than $L + k$ if one sticks to a statistically hiding commitment scheme.

and that the whole recursive process is computed honestly. As a result, their witness sizes are all much larger than the optimal size $\ell + 2\ell n$.

ZK of a Plaintext. We now introduce a ZK argument of knowledge of a plaintext for a variant of McEliece encryption scheme [63,70], where the noise is a regular word. Let $\mathbf{G} \in \mathbb{F}_2^{n_e \times k_e}$ be the public key and $\mathbf{c} \in \mathbb{F}_2^{n_e}$ be a ciphertext, k_1, k_2, k be positive integers such that $k_1 + k_2 = k_e$ and $\frac{k}{c} \cdot 2^c = n_e$. The prover is to prove the knowledge of $\mathbf{u} \in \mathbb{F}_2^{k_1}$, $\mathbf{m} \in \mathbb{F}_2^{k_2}$ as well as $\mathbf{e}' \in \mathbb{F}_2^k$ such that

$$\mathbf{c} = \mathbf{G} \cdot \begin{pmatrix} \mathbf{u} \\ \mathbf{m} \end{pmatrix} + \text{RE}(\mathbf{e}'). \quad (3)$$

Similarly, we prove that $\mathbf{e} = \text{RE}(\mathbf{e}')$ is a regular word by demonstrating that \mathbf{e} is the correct regular encoding of some vector \mathbf{e}' . In addition, proving the knowledge of vectors \mathbf{u}, \mathbf{m} is straightforward since we can view them as the identity function on $\{0, 1\}^{k_e}$.

ZK for Advanced Privacy-preserving Primitives. Being prepared with the above ZK protocols for various code-based relations, we are able to design ZK protocols for RS scheme [69], GS scheme [69], and FDABS scheme [56]. In particular, the ZK for RS scheme is an extension of the ZK for proving the regular encoding process and for proving an accumulated value. The ZK for GS scheme is then an extension of the ZK for RS by incorporating the ZK for proving the knowledge of a plaintext for the above variant of McEliece encryption. Finally, the ZK for FDABS scheme is an extension of ZK protocols for proving an accumulated value and for proving valid opening by incorporating a ZK for circuit satisfiability as well as a ZK for proving an odd-weight vector.

2 Preliminaries

Notations. Let λ be the security parameter. We use $x \stackrel{\$}{\leftarrow} S$ to denote the process of sampling x uniformly at random from a finite set S . Let $[a, b) := \{a, \dots, b-1\}$ and we often write $[1, b]$ as $[b]$. Let \oplus denote the bitwise exclusive-or. For a bit j , let $\bar{j} = j \oplus 1$. Throughout this paper, all vectors are column vectors and represented by bold lowercase letters (e.g., \mathbf{x}). Denote by x_i and $\mathbf{x}_{[i,j]}$ the i -component of vector \mathbf{x} and the vector consisting of x_i, x_{i+1}, \dots, x_j . Let $(\mathbf{x} \parallel \mathbf{y}) \in \mathbb{F}_2^{m+n}$ and $[\mathbf{A} \mid \mathbf{B}] \in \mathbb{F}_2^{n \times (m+k)}$ be the concatenation of vectors $\mathbf{x} \in \mathbb{F}_2^m$ and $\mathbf{y} \in \mathbb{F}_2^n$, and matrices $\mathbf{A} \in \mathbb{F}_2^{n \times m}$ and $\mathbf{B} \in \mathbb{F}_2^{n \times k}$. For an integer $j \in [0, 2^\ell - 1]$, denote its binary representation by $\text{bin}(j) \in \{0, 1\}^\ell$.

2.1 Boolean Functions

Let $f \in \mathbb{F}_2[X_1, \dots, X_c]$ be a c -variate Boolean function: $\mathbb{F}_2^c \rightarrow \mathbb{F}_2$. Then a representation of f is by its truth table, i.e.,

$$\text{TT}(f) = [f(0, 0, \dots, 0), f(0, \dots, 0, 1), \dots, f(0, 1, \dots, 1), f(1, 1, \dots, 1)].$$

Clearly, the representation is unique. It is known (see e.g., [28,71]) that any Boolean function f in c variables can be expressed in terms of a multivariate polynomial in $\mathbb{F}_2[X_1, X_2, \dots, X_c]/(X_1^2 + X_1, X_2^2 + X_2, \dots, X_c^2 + X_c)$:

$$f(X_1, X_2, \dots, X_c) = \sum_{\mathbf{u} \in \mathbb{F}_2^c} a_{\mathbf{u}} \left(\prod_{j=1}^c X_j^{u_j} \right) = \sum_{\mathbf{u} \in \mathbb{F}_2^c} a_{\mathbf{u}} \mathbf{X}^{\mathbf{u}},$$

where $\mathbf{X} = (X_1, X_2, \dots, X_c)$, $\mathbf{u} = (u_1, u_2, \dots, u_c) \in \mathbb{F}_2^c$, $a_{\mathbf{u}} \in \mathbb{F}_2$ and the term $\mathbf{X}^{\mathbf{u}} = \prod_{i=1}^c X_i^{u_i}$ is called a monomial. This representation is called the algebraic normal form (ANF) of f . The algebraic degree of f , denoted by $\text{deg}(f)$, is then defined as the maximum value of $\text{wt}(\mathbf{u})$ with $a_{\mathbf{u}} \neq 0$.

2.2 Code-Based Collision Resistant Hash Functions

Augot, Finiasz and Sendrier (AFS) [4,5] introduced regular syndrome decoding (RSD) and 2-regular null syndrome decoding (2-RNSD) problems and proposed a family of code-based hash functions based on the hardness of the latter problem. Later, Nguyen et al. [69] developed the AFS hash function to obtain code-based computationally binding and statistically hiding commitment scheme. We first provide some related notions following [69] and then recall the AFS hash functions.

Let k, c be positive integers and c divides k . Define the following.

Regular(k, c) is the set of all vectors $\mathbf{y} = (\mathbf{y}_1 \parallel \dots \parallel \mathbf{y}_{k/c}) \in \mathbb{F}_2^{k/c \cdot 2^c}$ consisting of k/c blocks, each of which is a unit vector of length 2^c . We call \mathbf{y} a *regular word* if $\mathbf{y} \in \text{Regular}(k, c)$ for some k, c .

RE : $\mathbb{F}_2^k \rightarrow \mathbb{F}_2^{k/c \cdot 2^c}$ is a regular encoding function that encodes $\mathbf{x} = (\mathbf{x}_1 \parallel \dots \parallel \mathbf{x}_{k/c}) \in \mathbb{F}_2^k$ to $\mathbf{y} = \text{RE}(\mathbf{x}) = (\mathbf{y}_1 \parallel \dots \parallel \mathbf{y}_{k/c})$. In particular, for $\mathbf{x}_j = (x_{j,1}, \dots, x_{j,c})^\top$, let $t_j = \sum_{k=1}^c x_{j,k} \cdot 2^{c-k} \in [0, 2^c - 1]$ be the integer represented by \mathbf{x}_j . Then \mathbf{y}_j is the unit vector of length 2^c that has the sole 1 at position $t_j + 1$. It is straightforward to see that $\mathbf{y} \in \text{Regular}(k, c)$.

2-Regular(k, c) is the set of all vectors $\mathbf{x} \in \mathbb{F}_2^{k/c \cdot 2^c}$ such that exist regular words $\mathbf{v}, \mathbf{w} \in \text{Regular}(k, c)$ satisfying $\mathbf{x} = \mathbf{v} \oplus \mathbf{w}$. Notice that $x \in \text{2-Regular}(k, c)$ if and only if it can be written as the concatenation of k/c blocks of length 2^c , each of which has Hamming weight 0 or 2. We call \mathbf{x} a *2-regular word* if $\mathbf{x} \in \text{2-Regular}(k, c)$ for some k, c .

RSD and 2-RNSD problems are variants of the famous SD problem, in which the goals are to find regular words and 2-regular words. As proved in [4,5], both problems are NP-complete. We recall them below.

Definition 1 (Regular Syndrome Decoding Problem). *Let n, k, c be three positive integers, $n > c$, and $k/c \cdot 2^c > k$. Define $m = k/c \cdot 2^c$. Given a uniform random matrix $\mathbf{B} \in \mathbb{F}_2^{n \times m}$, the regular syndrome decoding $\text{RSD}_{n,k,c}$ problem asks to find a $\mathbf{x} \in \mathbb{F}_2^k$ such that $\mathbf{B} \cdot \text{RE}(\mathbf{x}) = \mathbf{0} \pmod 2$.*

The Hardness of RSD Problem. A number of works have analyzed the hardness of RSD problem under different parameter regimes, e.g., [40, 48, 59]. In particular, some recent works [24, 29, 38] have utilized the regular noise structure into account, resulting in better algebraic attacks for RSD problem. As shown in [58, Table 8], such attacks work better than other pooled Gauss attack [37] or information set decoding (ISD) attack [72] for RSD with low-noise weight. Looking ahead, we work with parameters where the exact relations between RSD and SD problems remains unclear [29, 38]. Therefore, we follow the approach presented in [29] to select parameters.

Definition 2 (2-Regular Null Syndrome Decoding Problem). *Let n, k, c be three positive integers, $n > c$, and $k/c \cdot 2^c > k$. Define $m = k/c \cdot 2^c$. Given a uniform random matrix $\mathbf{B} \in \mathbb{F}_2^{n \times m}$, the 2-regular null syndrome decoding 2-RNSD $_{n,k,c}$ problem asks to find a $\mathbf{z} \in 2\text{-Regular}(k, c)$ such that $\mathbf{B} \cdot \mathbf{z} = \mathbf{0} \pmod{2}$.*

Note that 2-RNSD problem is equivalent to finding two different $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^k$ such that $\mathbf{B} \cdot \text{RE}(\mathbf{x}) = \mathbf{B} \cdot \text{RE}(\mathbf{y})$.

The Hardness of 2-RNSD Problem. Augot et al. [4, 5] applied ISD attack and generalized birthday attack (GBA) [78] to 2-RNSD problem, as well as giving lower bound on the cost of those two attacks. Later, Augot et al. [3] improved upon previous results and proposed several parameters for achieving different security levels. Follow-up works [15–17] proposed further improvements. As explicitly stated in [15, 16], however, the parameters chosen in [3] are too conservative so that the further improved algorithms [15–17] do not violate the security claims made by Augot et al. [3]. To this end, we choose parameters for 2-RNSD problem according to [3].

The AFS Hash Functions. Let $n, k = \Omega(\lambda)$, $k > n$, and $c|k$. The AFS family of hash functions, specified by parameters n, k, c , is the set $\{h_{\mathbf{B}} : \mathbb{F}_2^k \rightarrow \mathbb{F}_2^n, \mathbf{B} \in \mathbb{F}_2^{n \times 2^c \cdot k/c}\}$ that maps \mathbf{x} to $\mathbf{B} \cdot \text{RE}(\mathbf{x}) \pmod{2}$.

It is straightforward to see that the above hash functions are collision-resistant based on the hardness of the 2-RNSD $_{n,k,c}$ problem.

The Modified AFS Hash Function. Nguyen et al. [69] recently modified the AFS hash function family [5] so that it takes 2 inputs (instead of just 1) and hence is suitable for building Merkle hash trees. The definition is given below.

Definition 3. *Let $m = 2 \cdot 2^c \cdot n/c$. The function family \mathcal{H} mapping $\mathbb{F}_2^n \times \mathbb{F}_2^n$ to \mathbb{F}_2^n is defined as $\mathcal{H} = \{h_{\mathbf{B}} \mid \mathbf{B} \in \mathbb{F}_2^{n \times m}\}$, where for $\mathbf{B} = [\mathbf{B}_0 \mid \mathbf{B}_1]$ with $\mathbf{B}_0, \mathbf{B}_1 \in \mathbb{F}_2^{n \times m/2}$, and for any $(\mathbf{u}_0, \mathbf{u}_1) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, we have:*

$$h_{\mathbf{B}}(\mathbf{u}_0, \mathbf{u}_1) = \mathbf{B}_0 \cdot \text{RE}(\mathbf{u}_0) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{u}_1) \in \mathbb{F}_2^n.$$

The collision resistance of the hash function family relies on the hardness of the 2-RNSD $_{n,2n,c}$ problem [69].

2.3 Code-Based Commitment Scheme

The above AFS hash functions can be used to build a commitment scheme. We now recall the statistically hiding and computationally binding commitment scheme proposed in [69].

CSetup(1^λ): Given the security parameter 1^λ , it chooses $n = \mathcal{O}(\lambda)$, $k \geq n + 2\lambda + \mathcal{O}(1)$, and specifies the message space $\mathcal{X} = \mathbb{F}_2^L$. It also chooses $c = \mathcal{O}(1)$ that divides both k and L . Let $m_0 = 2^c \cdot L/c$ and $m_1 = 2^c \cdot k/c$. Sample $\mathbf{C}_0 \xleftarrow{\$} \mathbb{F}_2^{n \times m_0}$ and $\mathbf{C}_1 \xleftarrow{\$} \mathbb{F}_2^{n \times m_1}$. Output public parameter $\mathbf{pp} = \{\lambda, n, k, L, c, m_0, m_1, \mathbf{C}_0, \mathbf{C}_1\}$.

CCom(\mathbf{pp}, \mathbf{x}): To commit to a message $\mathbf{x} \in \mathbb{F}_2^L$, this algorithm samples a randomness $\mathbf{r} \xleftarrow{\$} \mathbb{F}_2^{n \times k}$, computes $\mathbf{c} = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{r})$, and outputs commitment \mathbf{c} as well as the opening \mathbf{r} .

COpen($\mathbf{pp}, \mathbf{c}, (\mathbf{x}, \mathbf{r})$): Given the inputs, it outputs 1 if $\mathbf{c} = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{r})$ and 0 otherwise.

Lemma 1 ([69]). *The above commitment scheme is correct. For any $\mathbf{x} \in \mathbb{F}_2^L$, the distribution of commitment \mathbf{c} is statistically close to the uniform distribution over \mathbb{F}_2^n . In particular, the scheme satisfies the statistical hiding property. Moreover, if 2-RNSD $_{n,L+k,c}$ problem is hard, then the scheme is also computationally binding.*

2.4 Updatable Code-Based Merkle-Tree Accumulator

We now recall the updatable code-based Merkle-tree accumulator [68,69].

TSetup(1^λ). This algorithm first chooses $n = \mathcal{O}(\lambda)$, $c = \mathcal{O}(1)$ so that c divides n . Set $m = 2 \cdot 2^c \cdot n/c$. It then samples $\mathbf{B} \xleftarrow{\$} \mathbb{F}_2^{n \times m}$, and outputs the public parameter $\mathbf{pp} = \{\lambda, n, c, m, \mathbf{B}\}$.

TAcc($R = \{\mathbf{d}_0, \dots, \mathbf{d}_{N-1}\} \subseteq (\mathbb{F}_2^n)^N$). Assume $N = 2^\ell$ without loss of generality. Re-write \mathbf{d}_j as $\mathbf{u}_{\ell,j}$ and call \mathbf{d}_j the leaf value of the leaf node $\text{bin}(j)$ for $j \in [0, N - 1]$. Build a binary tree upon N leaves $\mathbf{u}_{\ell,0}, \dots, \mathbf{u}_{\ell,2^\ell-1}$ in the following way. For $k \in \{\ell - 1, \ell - 2, \dots, 1, 0\}$ and $i \in [0, 2^k - 1]$, compute $\mathbf{u}_{k,i} = h_{\mathbf{B}}(\mathbf{u}_{k+1,2i}, \mathbf{u}_{k+1,2i+1})$. Output the accumulated value $\mathbf{u} = \mathbf{u}_{0,0}$.

TWitGen(R, \mathbf{d}). If $\mathbf{d} \notin R$, the algorithm outputs \perp . Otherwise, it outputs the witness w for \mathbf{d} as follows.

1. Set $\mathbf{d} = \mathbf{d}_j$ for some $j \in [0, N - 1]$. Re-write $\mathbf{d}_j = \mathbf{u}_{\ell,j}$. Let $\text{bin}(j) = (j_1, \dots, j_\ell)^\top \in \{0, 1\}^\ell$ be the binary representation of j .
2. Consider the path from $\mathbf{u}_{\ell,j}$ to the root \mathbf{u} , the witness w then consists of $\text{bin}(j)$ as well as all the sibling nodes of the path. Let $w = (\text{bin}(j), (\mathbf{w}_\ell, \dots, \mathbf{w}_1)) \in \mathbb{F}_2^\ell \times (\mathbb{F}_2^n)^\ell$.

TVerify($\mathbf{u}, \mathbf{d}, w$). Let w be of the following form:

$$w = ((j_1, \dots, j_\ell)^\top, (\mathbf{w}_\ell, \dots, \mathbf{w}_1)).$$

This algorithm then computes $\mathbf{v}_\ell, \dots, \mathbf{v}_0$. Let $\mathbf{v}_\ell = \mathbf{d}$ and

$$\forall i \in \{\ell - 1, \dots, 1, 0\} : \mathbf{v}_i = \begin{cases} h_{\mathbf{B}}(\mathbf{v}_{i+1}, \mathbf{w}_{i+1}), & \text{if } j_{i+1} = 0; \\ h_{\mathbf{B}}(\mathbf{w}_{i+1}, \mathbf{v}_{i+1}), & \text{if } j_{i+1} = 1. \end{cases} \quad (4)$$

Output 1 if $\mathbf{v}_0 = \mathbf{u}$ or 0 otherwise.

TUpdate(bin(j), \mathbf{d}^*): Let \mathbf{d}_j be the existing leaf value of the leaf node $\text{bin}(j)$. It executes the algorithm $\text{TWitGen}_{\mathbf{B}}(R, \mathbf{d}_j)$, obtaining $w = (\text{bin}(j), (\mathbf{w}_\ell, \dots, \mathbf{w}_1))$. It then sets $\mathbf{v}_\ell = \mathbf{d}^*$ and recursively computes $\mathbf{v}_{\ell-1}, \dots, \mathbf{v}_0$ as in (4). Finally, for $i \in [0, \ell]$, it sets $\mathbf{u}_{i, \lfloor \frac{j}{2^{\ell-i}} \rfloor} = \mathbf{v}_i$.

Lemma 2 ([69]). *Assume that the $2\text{-RNSD}_{n,2n,c}$ problem is hard, then the given accumulator scheme is correct and secure, i.e., it is infeasible to prove that a value \mathbf{d}^* was accumulated in a value \mathbf{u} if it was not (see, e.g., [27, 54] for formal definition).*

2.5 Randomized McEliece Encryption Schemes

Now we recall a randomized variant of the McEliece [63] encryption scheme as suggested in [70].

ME.Setup(1^λ). Let $n_e = n_e(\lambda)$, $k_e = k_e(\lambda)$, $t_e = t_e(\lambda)$ be the parameters for a binary $[n_e, k_e, 2t_e + 1]$ Goppa code. Choose $k_1, k_2 \in \mathbb{Z}$ such that $k_e = k_1 + k_2$. Let $\mathbb{F}_2^{k_2}$ be the plaintext space.

ME.KeyGen(n_e, k_e, t_e). This algorithm outputs the encryption key and decryption key for the randomized McEliece encryption scheme. It works as follows:

1. Choose a generator matrix $\mathbf{G}' \in \mathbb{F}_2^{n_e \times k_e}$ of a random $[n_e, k_e, 2t_e + 1]$ Goppa code. Let $\mathbf{S} \in \mathbb{F}_2^{k_e \times k_e}$ be a random invertible matrix and $\mathbf{P} \in \mathbb{F}_2^{n_e \times n_e}$ be a random permutation matrix, compute $\mathbf{G} = \mathbf{P}\mathbf{G}'\mathbf{S} \in \mathbb{F}_2^{n_e \times k_e}$.
2. Output encryption key $\text{pk}_{\text{ME}} = \mathbf{G}$ and decryption key $\text{sk}_{\text{ME}} = (\mathbf{S}, \mathbf{G}', \mathbf{P})$.

ME.Enc($\text{pk}_{\text{ME}}, \mathbf{m}$). On input a message $\mathbf{m} \in \mathbb{F}_2^{k_2}$ and pk_{ME} , sample random $\mathbf{u} \xleftarrow{\$} \mathbb{F}_2^{k_1}$ and $\mathbf{e} \in \mathbb{F}_2^{n_e}$ such that the Hamming weight of \mathbf{e} is exactly t_e , and then output the ciphertext $\mathbf{c} = \mathbf{G} \cdot \begin{pmatrix} \mathbf{u} \\ \mathbf{m} \end{pmatrix} \oplus \mathbf{e} \in \mathbb{F}_2^{n_e}$.

ME.Dec($\text{sk}_{\text{ME}}, \mathbf{c}$). On input the ciphertext \mathbf{c} and decryption key sk_{ME} , it works as follows:

1. Multiply \mathbf{P}^{-1} to the left of the ciphertext \mathbf{c} , then apply an error-correcting algorithm. Obtain $\mathbf{m}'' = \text{Decode}_{\mathbf{G}'}(\mathbf{c} \cdot \mathbf{P}^{-1})$, where Decode is an error-correcting algorithm with respect to \mathbf{G}' . Returns \perp if Decode fails.
2. Multiply \mathbf{S}^{-1} to the right of the ciphertext \mathbf{m}'' , then $\mathbf{m}' = \mathbf{S}^{-1} \cdot \mathbf{m}''$, parse $\mathbf{m}' = \begin{pmatrix} \mathbf{u} \\ \mathbf{m} \end{pmatrix}$, where $\mathbf{u} \in \mathbb{F}_2^{k_1}$ and $\mathbf{m} \in \mathbb{F}_2^{k_2}$. Return \mathbf{m} .

The above scheme is CPA-secure if it is infeasible to distinguish the matrix \mathbf{G} from random and the decisional learning parity with (exact) noise (DLPN) problem is computationally hard.

Definition 4 (Decisional Learning Parity with Exact Noise Problem).

Let N, k, t be three integers with $N > k$ and $N > t$. The decisional learning parity with (exact) noise $\text{DLPN}_{N,k,t}$ problem asks to distinguish if a given pair $(\mathbf{G}, \mathbf{r}) \in \mathbb{F}_2^{N \times k} \times \mathbb{F}_2^N$ is uniformly random or obtained by choosing $\mathbf{G} \xleftarrow{\$} \mathbb{F}_2^{N \times k}$, $\mathbf{s} \xleftarrow{\$} \mathbb{F}_2^k$, $\mathbf{e} \xleftarrow{\$} \mathbb{F}_2^t$ with exact Hamming weight t and then outputting $(\mathbf{G}, \mathbf{G} \cdot \mathbf{s} \oplus \mathbf{e})$.

A Variant with Regular Noise. In this work, we consider a variant of McEliece encryption scheme such that the noise \mathbf{e} is a regular word. More specifically, let k, c be two integers with $c|k$ such that $\frac{k}{c} \cdot 2^c = n_e$. Then the noise is computed as $\mathbf{e} = \text{RE}(\mathbf{e}')$ with $\mathbf{e}' \xleftarrow{\$} \mathbb{F}_2^k$. The security of this variant will then rely on the hardness of decisional LPN problem with regular noise, which is dual to the decisional RSD problem.

2.6 VOLE-Based Zero-Knowledge Proofs

Vector oblivious linear evaluation (VOLE). VOLE is a two-party functionality $\mathcal{F}_{\text{VOLE}}^{p,r}$ between a sender and receiver. It allows the sender to obtain $\mathbf{M} \in \mathbb{F}_{p^r}^l$ and $\mathbf{u} \in \mathbb{F}_p^l$ and the receiver to obtain $\mathbf{K} \in \mathbb{F}_{p^r}^l$ and $\Delta \in \mathbb{F}_{p^r}$ such that $\mathbf{K} = \mathbf{M} + \mathbf{u} \cdot \Delta$. These VOLE correlations can be used to authenticate \mathbf{u} . We denote such authenticated values by $\llbracket \mathbf{u} \rrbracket$, indicating that the sender obtains \mathbf{u} and \mathbf{M} while the receiver obtains Δ and \mathbf{K} . It is not hard to see that the sender cannot alter \mathbf{u} to a different \mathbf{u}' without guessing Δ correctly. It is also easy to verify that VOLE correlations are additively homomorphic. In particular, given public coefficients $c_0, \dots, c_l \in \mathbb{F}_{p^r}$, two parties can locally compute $\llbracket y \rrbracket = \sum_{i=1}^l c_i \cdot \llbracket u_i \rrbracket + c_0$, where the sender computes $y := \sum_{i=1}^l c_i \cdot u_i + c_0$ and $\mathbf{M}_y = \sum_{i=1}^l c_i \cdot \mathbf{M}_{u_i}$, and the receiver computes $\mathbf{K}_y := \sum_{i=1}^l c_i \cdot \mathbf{K}_{u_i} + c_0 \cdot \Delta$.

VOLE-Based ZK Proofs. A VOLE-based ZK protocol for circuit satisfiability works in two phases. First, two parties call the functionality $\mathcal{F}_{\text{VOLE}}^{p,r}$ to obtain random VOLE correlations. Using these correlations, the two parties obtain VOLE correlations for all wire values. This is done by letting \mathcal{P} commit to all input wire values and output wire values of multiplication gates. Due to the homomorphic property of VOLE, they will also obtain VOLE correlations for the output wire values of addition gates. Next, they run subprotocols to check that all multiplications gates are computed honestly. One approach proposed by Ditter et al. [35] and later improved by Yang et al. [82] employs the fact that VOLE correlations are linear relationships, and works as follows.

For i -th multiplication gate, \mathcal{P} has $(\mathbf{M}_1, w_1), (\mathbf{M}_2, w_2), (\mathbf{M}_3, w_3) \in \mathbb{F}_p \times \mathbb{F}_{p^r}$, and the verifier \mathcal{V} possesses $\Delta, \mathbf{K}_1, \mathbf{K}_2, \mathbf{K}_3 \in \mathbb{F}_{p^r}$ such that

$$w_3 = w_1 \cdot w_2, \quad \text{and} \quad \mathbf{K}_i = \mathbf{M}_i + w_i \cdot \Delta \quad \text{for} \quad i \in \{1, 2, 3\}. \tag{5}$$

If the circuit is computed correctly, then

$$\begin{aligned}
 B_i &= \underbrace{K_1 \cdot K_2 - K_3 \cdot \Delta}_{\text{known to } \mathcal{V}} \\
 &= \underbrace{M_1 \cdot M_2}_{\text{known to } \mathcal{P}} + \underbrace{(M_2 \cdot w_1 + M_1 \cdot w_2 - M_3)}_{\text{known to } \mathcal{P}} \cdot \Delta + \underbrace{(w_1 \cdot w_2 - w_3)}_{0 \text{ if } \mathcal{P} \text{ is honest}} \cdot \Delta^2 \\
 &= A_{i,0} + A_{i,1} \cdot \Delta.
 \end{aligned} \tag{6}$$

Therefore, checking the quadratic constraints of multiplication gates can be converted to checking the above linear Eq. (6). Moreover, we can use random linear combination to reduce checking t equations (corresponding to t multiplication gates) to checking a single equation. More specifically, the verifier \mathcal{V} samples a uniform vector $\chi \in \mathbb{F}_{p^r}^t$ and sends it to \mathcal{P} , who returns back $A_0 = \sum_{i=1}^t \chi_i \cdot A_{i,0} + A_0^*$ and $A_1 = \sum_{i=1}^t \chi_i \cdot A_{i,1} + A_1^*$. The verifier then check if $\sum_{i=1}^t \chi_i \cdot B_i + B^* = A_0 + A_1 \cdot \Delta$. Here $B^* = A_0^* + A_1^* \cdot \Delta$ is another random VOLE correlation.

Vector Oblivious Polynomial Evaluation (VOPE). VOPE, first introduced by Yang et al. [82], is an extension of VOLE, in which the sender gets $A_0, \dots, A_d \in \mathbb{F}_{p^r}$ while the receiver gets $B \in \mathbb{F}_{p^r}$ and $\Delta \in \mathbb{F}_{p^r}$ such that $B = \sum_{i \in [0,d]} A_i \cdot \Delta^i$. Such VOPE correlations are particularly efficient for proving polynomial satisfiability. As shown in [82], it is possible to prove a set of degree- d polynomials on totally l distinct variables with communication cost of $l + d$ field elements, which is independent of the number of multiplications to compute all polynomials. Let f_1, \dots, f_t be a set of l -variate degree- d polynomials over \mathbb{F}_{p^k} . For simplicity, we represent each polynomial in a degree-separated format, i.e., $f_i(X_1, \dots, X_l) = \sum_{h \in [0,d]} g_{i,h}(X_1, \dots, X_l)$ such that all terms in $g_{i,h}$ have degree exactly h . The prover wants to prove that $f_i(w_1, \dots, w_l) = 0$ for $i \in [1, t]$ with $\mathbf{w} = (w_1, \dots, w_l)^\top \in \mathbb{F}_p^l$. Similar to the VOLE-based ZK, \mathcal{P} first commits to the witness \mathbf{w} , and then checks that all polynomials are satisfied. The key observation is that one can obtain a degree- $(d - 1)$ constraint generalized from (6). In more detail, suppose \mathcal{P} and \mathcal{V} obtain $[[w_1]], \dots, [[w_l]]$ such that $K_i = M_i + w_i \cdot \Delta$. Then

$$\begin{aligned}
 B_i &= \underbrace{\sum_{h=0}^d g_{i,h}(K_1, \dots, K_l) \cdot \Delta^{d-h}}_{\text{known to } \mathcal{V}} = \sum_{h=0}^d g_{i,h}(M_1 + w_1 \cdot \Delta, \dots, M_l + w_l \cdot \Delta) \cdot \Delta^{d-h} \\
 &= \underbrace{f(w_1, \dots, w_n)}_{0 \text{ if } \mathcal{P} \text{ is honest}} \cdot \Delta^d + \underbrace{A_{i,0}}_{\text{known to } \mathcal{P}} + \underbrace{A_{i,1}}_{\text{known to } \mathcal{P}} \cdot \Delta + \dots + \underbrace{A_{i,d-1}}_{\text{known to } \mathcal{P}} \cdot \Delta^{d-1}. \tag{7}
 \end{aligned}$$

Finally, utilizing the random linear combination technique and a degree- $(d - 1)$ VOPE correlation, one reduces checking t equations to checking a single equation.

2.7 VOLE-In-The-Head

A main drawback of the above VOLE-based and VOPE-based ZK proof systems is that of being inherently designated-verifier (DV) since \mathcal{V} has to know its part of VOLE/VOPE correlations so as to verify the proofs. We now briefly recall the VOLE-in-the-head (VOLEitH) technique presented by Baum et al. [8] that transforms the above DVZK proofs to public-coin protocols, which in turn can be made non-interactive via the Fiat-Shamir heuristic [44].

At a high level, Baum et al. [8] employed a delayed VOLE functionality $\mathcal{F}_{\text{sVOLE}}^{p,q,S_\Delta,C,l,\mathcal{L}}$ that allows \mathcal{P} to first generate its values K_i, u_i independent of Δ, M_i and to generate Δ, M_i *after* all proof messages have been “committed”. This delayed VOLE functionality can then be realized via all-but-one oblivious transfer, which is further realized by GGM-based vector commitments (VC). Since in this work we focus on utilizing VOLEitH-based proof systems, we refrain from providing all the details about how to realize this delayed functionality. In the $\mathcal{F}_{\text{sVOLE}}^{p,q,S_\Delta,C,l,\mathcal{L}}$ -hybrid model, they then presented two instantiations. The first one allows one to prove statements over large fields, and the second one is more tailored for proving statements over small fields. In this work, we focus on their second instantiation.

Optimizations of VOLEitH. Though being a new paradigm, VOLEitH has received significant attention from the community [6, 18, 26, 32, 55] and several works have improved upon VOLEitH. Baum et al. [6] introduced batch all-but-one VC, rejection sampling, and proof of work at the prover’s side to reduce commitment opening sizes. Also, the improved GGM-based puncturable pseudo-random function proposed by Bui et al. [26] can be used as a drop-in replacement of the GGM-based VC. These optimizations are fundamental to improving the realization of the delayed functionality $\mathcal{F}_{\text{sVOLE}}^{p,q,S_\Delta,C,l,\mathcal{L}}$. While introducing significant improvement when designing standard signature schemes, these optimizations are relatively small for designing advanced primitives such as RS and GS. Nevertheless, in the corresponding sections, we will briefly mention how these optimizations improve the signature sizes of our constructions.

3 New Techniques for Proving Regular Encoding Process

In this section, we introduce new techniques for proving the correctness of a regular encoding process within the VOLEitH paradigm [8]. Our starting point is to explicitly express the non-linear regular encoding function RE as low-degree polynomial relations, which then can be proved efficiently using VOLEitH. To this end, we first present a protocol $\Pi_{\text{dD-Rep}}^t$ for proving degree- d polynomial constraints in Sect. 3.1, which is a generalization of the protocol $\Pi_{\text{2D-Rep}}^t$ [8] for proving degree-2 constraints. Then, we show how to express the non-linear regular encoding function RE as low-degree polynomial relations in Sect. 3.2.

3.1 VOLE-In-The-Head for Degree- d Constraints

Let us now describe our protocol $\Pi_{\text{dD-Rep}}^t$, which is a generalization of the protocol $\Pi_{\text{2D-Rep}}^t$ [8] by incorporating the techniques in QuickSilver [82] for proving degree- d polynomial satisfiability. The goal of \mathcal{P} is to prove the knowledge of $\mathbf{w} \in \mathbb{F}_p^l$ such that $f_i(\mathbf{w}) = 0$ for $i \in [1, t]$, where $\{f_1, \dots, f_t\}$ are l -variate degree- d polynomials. The protocol follows the commit-and-prove paradigm and works as follows.

In the commit phase, both parties invoke the delayed functionality $\mathcal{F}_{\text{SVOLE}}$. The prover obtains $\mathbf{u} \in \mathbb{F}_p^{l+(d-1)r\tau}$ and \mathbf{V} , while the verifier \mathcal{V} will obtain \mathbf{Q} and Δ satisfying $\mathbf{Q} = \mathbf{V} + \mathbf{u} \cdot \mathbf{G}_C \text{diag}(\Delta)$ (after receiving messages from \mathcal{P} in the prove phase). Next, \mathcal{P} commits to its witness \mathbf{w} by sending $\mathbf{d} = \mathbf{w} - \mathbf{u}_{[1,l]}$ to \mathcal{V} .

In the challenge phase, \mathcal{V} samples uniformly random coefficients χ_1, \dots, χ_t and sends them to \mathcal{P} . These coefficients will be used for the random linear combination performed by \mathcal{P} in the following phase.

In the prove phase, \mathcal{P} basically reduces the task of proving $f_i(\mathbf{w}) = 0$ for $i \in [1, t]$ into the task of proving the satisfiability of (7). In the process, both parties employ the remaining $(d-1)r\tau$ relations related to $\mathbf{u}_{[l+1, l+(d-1)r\tau]}$ to generate a single VOPE correlation so as to mask a random linear combination of the t Eq. (7). Note that $(d-1)r\tau$ relations are required here while $(2d-1)r\tau$ are needed in QuickSilver. As pointed out by [6], this is because that QuickSilver VOPE generation should be secure against malicious verifier while VOLEitH does not have to.

Details of the generalization are given in the protocol $\Pi_{\text{dD-Rep}}^t$.

Protocol 1: $\Pi_{\text{dD-Rep}}^t$

PARAMETERS: Code $\mathcal{C}_{\text{Rep}} = [\tau, 1, \tau]_p$ with $\mathbf{G}_C = (1, \dots, 1) \in \mathbb{F}_p^{1 \times \tau}$. $q = p^r$. Assume there is one-to-one correspondence between elements in \mathbb{F}_q and $[1, q]$. Define $S_\Delta = \mathbb{F}_q^\tau$.

INPUTS: Polynomials $f_i = \sum_{h \in [0, d]} f_{i,h} \in \mathbb{F}_{p^k}[X_1, \dots, X_l]_{\leq d}$, $i \in [t]$ with $k | (r\tau)$. \mathcal{P} holds a witness $\mathbf{w} = (w_1, \dots, w_l)^\top \in \mathbb{F}_p^l$ such that $f_i(w_1, \dots, w_l) = 0$ for all $i \in [t]$.

Round 1. \mathcal{P} performs the following steps.

1. Call the functionality $\mathcal{F}_{\text{SVOLE}}^{p,q,S_\Delta,\mathcal{C}_{\text{Rep}},l+(d-1)r\tau,\mathcal{L}}$ and receive $\mathbf{u} \in \mathbb{F}_p^{l+(d-1)r\tau}$, $\mathbf{V} \in \mathbb{F}_q^{(l+(d-1)r\tau) \times \tau}$. \mathcal{V} receives done.
2. Compute $\mathbf{d} = \mathbf{w} - \mathbf{u}_{[1,l]} \in \mathbb{F}_p^l$ and send \mathbf{d} to \mathcal{V} .
3. For $i \in [l+1, l+(d-1)r\tau]$, embed the i -th element $u_i \in \mathbb{F}_p$ of \mathbf{u} to $u_i \in \mathbb{F}_{q^\tau}$. For $i \in [l+(d-1)r\tau]$, lift the i -th row $\mathbf{v}_i \in \mathbb{F}_q^\tau$ of \mathbf{V} into $v_i \in \mathbb{F}_{q^\tau}$. For $i \in [l]$, also embed the i -th element w_i of witness \mathbf{w} to $w_i \in \mathbb{F}_{q^\tau}$.

Round 2. \mathcal{V} samples uniformly random $\chi_i \xleftarrow{\$} \mathbb{F}_{q^\tau}$, $i \in [t]$ and sends to \mathcal{P} .

Round 3. After receiving χ_1, \dots, χ_t , \mathcal{P} does the following.

1. For each $i \in [t]$, compute $A_{i,0}, A_{i,1}, \dots, A_{i,d-1} \in \mathbb{F}_{q^\tau}$ such that

$$c_i(Y) = \sum_{h=0}^d \overline{f_{i,h}}(v_1+w_1Y, \dots, v_l+w_lY)Y^{d-h} = \overline{f_i}(w_1, \dots, w_l) \cdot Y^d + \sum_{j=0}^{d-1} A_{i,j} \cdot Y^j,$$

where $\overline{f_{i,h}} \in \mathbb{F}_{q^\tau}[X_1, \dots, X_l]$ is the embedding of $f_{i,h} \in \mathbb{F}_{p^k}[X_1, \dots, X_l]$.

2. **Generation of a VOPE correlation.**

- a) For $j \in [1, d]$, compute

$$u_j^* = \sum_{i \in [r\tau]} u_{l+(j-1)r\tau+i} X^{i-1} \in \mathbb{F}_{q^\tau}, \quad v_j^* = \sum_{i \in [r\tau]} v_{l+(j-1)r\tau+i} X^{i-1} \in \mathbb{F}_{q^\tau}.$$

where $\mathbb{F}_{q^\tau} \cong \mathbb{F}_p[X]/F(X)$ with $F(X) \in \mathbb{F}_p[X]$ being an irreducible polynomial of degree $r\tau$.

- b) Define $g_1(x) = v_1^* + u_1^* \cdot x$. For $i \in [1, d-2]$, compute $g_{i+1}(x) = g_i(x)(v_{i+1}^* + u_{i+1}^* \cdot x)$. Then \mathcal{P} is able to compute the coefficients $A_0^*, \dots, A_{d-1}^* \in \mathbb{F}_{q^\tau}$ such that $g_{d-1}(x) = \sum_{j=0}^{d-1} A_j^* \cdot x^j$.

3. For $j \in [0, d)$, compute $\tilde{a}_j = \sum_{i \in [t]} \chi_i \cdot A_{i,j} + A_j^* \in \mathbb{F}_{q^\tau}$, and send \tilde{a}_j to \mathcal{V} .

Verification. After receiving all responses, \mathcal{V} runs the following checks.

1. Call $\mathcal{F}_{\text{SVOLE}}^{p,q,S_\Delta, \mathcal{C}_{\text{Rep}}, l+(d-1)r\tau, \mathcal{L}}$ on input (**get**) and obtain $\Delta \in S_\Delta, \mathbf{Q} \in \mathbb{F}_q^{(l+(d-1)r\tau) \times \tau}$ such that $\mathbf{Q} = \mathbf{V} + \mathbf{uG}_C \text{diag}(\Delta)$. Let \mathbf{q}_i be the i -th row vector of \mathbf{Q} , for $i \in [l+1, l+(d-1)r\tau]$.
2. Compute $\mathbf{Q}^* = \mathbf{Q}_{[1,l]} + \mathbf{d} \cdot \mathbf{G}_C \cdot \text{diag}(\Delta)$, which is supposed to be $\mathbf{V}_{[1,l]} + \mathbf{w} \cdot \mathbf{G}_C \cdot \text{diag}(\Delta)$. Let $\mathbf{q}_1^*, \dots, \mathbf{q}_l^* \in \mathbb{F}_q^\tau$ be the rows of \mathbf{Q}^* .
3. Lift $\Delta \in \mathbb{F}_q^\tau$ into $\Delta \in \mathbb{F}_{q^\tau}$. Also, lift $\mathbf{q}_1^*, \dots, \mathbf{q}_l^*, \mathbf{q}_{l+1}, \dots, \mathbf{q}_{l+(d-1)r\tau} \in \mathbb{F}_q^\tau$ into $q_1^*, \dots, q_l^*, q_{l+1}, \dots, q_{l+(d-1)r\tau} \in \mathbb{F}_{q^\tau}$.
4. **Generation of a VOPE correlation.**
 - a) For $j \in [1, d]$, compute $q_{l+j}^* = \sum_{i \in [r\tau]} q_{l+(j-1)r\tau+i} X^{i-1} \in \mathbb{F}_{q^\tau}$, which should satisfy $q_{l+j}^* = v_j^* + u_j^* \cdot \Delta$.
 - b) Let $B_1^* = q_{l+1}^*$. Then for $i \in [1, d-2]$, compute $B_{i+1}^* = B_i^* \cdot q_{l+i+1}^*$. Define $B^* = B_{d-1}^*$. Then one can verify that $B^* = \sum_{j=0}^{d-1} A_j^* \cdot \Delta^j$.
5. For each $i \in [t]$, compute

$$c_i(\Delta) = \sum_{h=0}^d \overline{f_{i,h}}(q_1^*, \dots, q_l^*) \cdot \Delta^{d-h}.$$

6. Compute $\tilde{c} = \sum_{i \in [t]} \chi_i \cdot c_i(\Delta) + B^*$ and check if $\tilde{c} = \sum_{j=0}^{d-1} \tilde{a}_j \cdot \Delta^j$.

Theorem 1. *The protocol $\Pi_{\text{dD-Rep}}^t$ realizes the functionality $\mathcal{F}_{\text{dD-ZK}}^t$ that proves degree- d polynomial satisfiability in the $\mathcal{F}_{\text{SVOLE}}^{p,q,S_\Delta, \mathcal{C}, l, \mathcal{L}}$ -hybrid model. The security holds against a malicious prover or a semi-honest verifier and the soundness error is bounded by $1/p^{r\tau} + d|S_\Delta|^{-1}$.*

Correctness of the protocol follows directly by inspection of the protocol. Details of simulation are given in the full version.

Communication Cost. In $\Pi_{\text{dD-Rep}}^t$, in addition to the cost of the sVOLE steps, \mathcal{P} sends the initial commitment $\mathbf{d} \in \mathbb{F}_p^l$ and $\{\tilde{a}_i \in \mathbb{F}_{q^\tau}\}_{i \in [0, d-1]}$. Therefore, the total cost is summarized as follows:

$$\text{Cost}_{\Pi_{\text{dD-Rep}}^t} = \text{Cost}_{\text{sVOLE}} + l \cdot \log_2 p + d \cdot r \cdot \tau \cdot \log_2 p. \quad (8)$$

Additionally, the verifier sends t values in \mathbb{F}_{q^τ} but this can be removed via the Fiat-Shamir transform in the non-interactive setting and does not affect the final proof size. When instantiating the delayed VOLE functionality $\mathcal{F}_{\text{sVOLE}}^{p,q,S_\Delta,C,l,\mathcal{L}}$ with the aforementioned GGM-based VC (see [8, Sect. 3.1, Fig. 3, Fig. 4] for details), the cost of sVOLE steps is

$$\begin{aligned} \text{Cost}_{\text{sVOLE}} &= 2\lambda + (l + (d-1) \cdot r \cdot \tau + h) \cdot (\tau - 1) \cdot \log_2 p \\ &\quad + (s + s \cdot \tau) \cdot \log_2 p + (2\lambda + r \cdot \lambda) \cdot \tau. \end{aligned} \quad (9)$$

The process of the instantiation employs an \mathbb{F}_p^l -hiding and ϵ -universal hash function $\mathbf{H} \in \mathbb{F}_p^{s \times (l + (d-1)r\tau + h)}$ (see the full version or [8, 74]). Looking ahead, when calculating concrete proof sizes in Sect. 5, we employ formulas (8) and (9).

Fiat-Shamir Transform. As shown by Baum et al. [8], applying the Fiat-Shamir transformation to $\Pi_{\text{dD-Rep}}^t$ (with the functionality $\mathcal{F}_{\text{sVOLE}}^{p,q,S_\Delta,C,l,\mathcal{L}}$ instantiated) results in a non-interactive zero-knowledge argument of knowledge. Ganesh et al. [45] also showed that the resulting non-interactive argument is simulation-extractable in the programmable random oracle model. At a high level, simulation-extractability guarantees that extractability holds even when the adversary sees simulated proofs, and it implies simulation-soundness [75]. Note that simulation-soundness is required for constructing CCA2-anonymous group signatures [13] that utilizes the Naor-Yung double encryption [67]. Therefore, replacing the Stern-like ZK protocol underlying the group signature scheme [69] with the above protocol will not degrade its security.

3.2 A New Technique for Proving the Regular Encoding Process

In this section, our target is to prove the regular encoding process within the VOLEitH paradigm. We then observe that it suffices to transform the regular encoding process into low-degree polynomial constraints. For simplicity, let us focus on the regular encoding function $\text{RE} : \mathbb{F}_2^c \rightarrow \mathbb{F}_2^{2^c}$.

We also observe that RE can be seen as 2^c number of c -variate Boolean functions $f_1(\cdot), \dots, f_{2^c}(\cdot)$. If we focus on the first output bit, then the truth table of the corresponding Boolean function $f_1(\cdot)$ is the unit vector $\mathbf{e}_1 \in \mathbb{F}_2^{2^c}$ with 1 in the first position. Through Lagrange interpolation, one can obtain $f_1(\cdot) \triangleq f_{(0,\dots,0)}(X_1, \dots, X_c) = \prod_{i=1}^c (1 + 0 + X_i)$. Interestingly, for the j -th output bit, the truth table of $f_j(\cdot)$ is the unit vector $\mathbf{e}_j \in \mathbb{F}_2^{2^c}$, and the Boolean function

is $f_j(\cdot) \triangleq f_{(j_1, \dots, j_c)}(X_1, \dots, X_c) = \prod_{i=1}^c (1 + j_i + X_i)$, where $(j_1, \dots, j_c)^\top = \text{bin}(j - 1)$.

To this end, we have successfully represented the non-linear encoding process as degree- c relations. In particular, $\text{RE}(x_1, \dots, x_c) =$

$$(f_{(0, \dots, 0)}(x_1, \dots, x_c), \dots, f_{(j_1, \dots, j_c)}(x_1, \dots, x_c), \dots, f_{(1, \dots, 1)}(x_1, \dots, x_c))^\top. \tag{10}$$

When applying RE to $\mathbf{x} \in \mathbb{F}_2^n$, we simply write $\text{RE}(\mathbf{x}) = (f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))^\top$ for $c|n$, $m = \frac{n}{c} \cdot 2^c$, and $\deg(f_i) = c$ for $i \in [1, m]$, without explicitly describing the details of f_i . In fact, $f_j(X_1, \dots, X_c) = f_{2^c+j}(X_1, \dots, X_c) = \dots = f_{(\frac{n}{c}-1) \cdot 2^c+j}(X_1, \dots, X_c)$ for $j \in [1, 2^c]$. Note that $f_j(\mathbf{x})$ only selects c inputs and ignores other inputs.

Therefore, to show that $\mathbf{z} = (z_1, \dots, z_m)^\top \in \text{Regular}(n, c)$ is indeed a regular encoding of $\mathbf{x} = (x_1, \dots, x_n)^\top \in \mathbb{F}_2^n$, it suffices to show that $z_j = f_j(\mathbf{x})$ for all $j \in [1, m]$. Since these m constraints are degree- c relations, and thus can be proved in zero-knowledge using the protocol $\Pi_{\text{dD-Rep}}^t$.

4 New Zero-Knowledge Protocols for Various Cryptographic Building Blocks

In this section, we provide new code-based zero-knowledge protocols that are essential for constructing privacy-enhancing primitives. This includes a ZK protocol for proving the knowledge of a committed value, a ZK protocol for proving the knowledge of a secret value that is accumulated honestly, and a ZK protocol for proving the knowledge of a plaintext for a variant of McEliece cryptosystem.

4.1 ZK of a Valid Opening

We first describe a ZK of a valid opening for the commitment scheme from Sect. 2.3. The goal of \mathcal{P} is to convince the verifier that it possesses witnesses $\mathbf{x} \in \mathbb{F}_2^L$ and $\mathbf{r} \in \mathbb{F}_2^k$ such that $\mathbf{c} = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{r})$. Denote $\mathbf{C}_0 = (c_{i,j})_{i \in [n], j \in [m_0]} \in \mathbb{F}_2^{n \times m_0}$ and $\mathbf{C}_1 = (c_{i,m_0+j})_{i \in [n], j \in [m_1]} \in \mathbb{F}_2^{n \times m_1}$ with $m_0 = \frac{L}{c} \cdot 2^c$ and $m_1 = \frac{k}{c} \cdot 2^c$. The protocol essentially relies on the techniques from Sect. 3.2 and works as follows.

Let $\tilde{\mathbf{x}} = (\mathbf{x}||\mathbf{r}) \in \mathbb{F}_2^{L+k}$, and $\mathbf{c} = (c_1, \dots, c_n)^\top$. Then $\mathbf{c} = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{r})$ is equivalent to $\mathbf{c} = [\mathbf{C}_0|\mathbf{C}_1] \cdot \text{RE}(\tilde{\mathbf{x}})$. Denote $\text{RE}(\tilde{\mathbf{x}}) = (f_1(\tilde{\mathbf{x}}), \dots, f_{m_0+m_1}(\tilde{\mathbf{x}}))^\top$. The prover then prepares n polynomials of degree c :

$$\phi_i(\cdot) = \sum_{j=1}^{m_0+m_1} c_{i,j} f_j(X_1, \dots, X_{L+k}) - c_i, \quad \forall i \in [n],$$

and the witness $\tilde{\mathbf{x}}$. At this point, \mathcal{P} runs the protocol $\Pi_{\text{dD-Rep}}^t$ and applies the Fiat-Shamir transform.

4.2 ZK of an Accumulated Value

Next, we describe a ZK of an accumulated value for the accumulator recalled in Sect. 2.4. The prover aims to prove knowledge of a hash chain from a secret leaf node to the root.

Specifically, the public inputs are $\mathbf{B} = [\mathbf{B}_0|\mathbf{B}_1] \in \mathbb{F}_2^{n \times m}$ and the root $\mathbf{u} \in \mathbb{F}_2^n$. The secret inputs consist of $(j_1, \dots, j_\ell)^\top \in \{0, 1\}^\ell$, $\mathbf{v}_1, \dots, \mathbf{v}_\ell \in \mathbb{F}_2^n$, and $\mathbf{w}_1, \dots, \mathbf{w}_\ell \in \mathbb{F}_2^m$ such that

$$\bar{j}_1 \cdot (\mathbf{B}_0 \cdot \text{RE}(\mathbf{v}_1) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{w}_1)) + j_1 \cdot (\mathbf{B}_0 \cdot \text{RE}(\mathbf{w}_1) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{v}_1)) = \mathbf{u}, \quad (11)$$

and for all $\theta \in [2, \ell]$:

$$\bar{j}_\theta \cdot (\mathbf{B}_0 \cdot \text{RE}(\mathbf{v}_\theta) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{w}_\theta)) + j_\theta \cdot (\mathbf{B}_0 \cdot \text{RE}(\mathbf{w}_\theta) \oplus \mathbf{B}_1 \cdot \text{RE}(\mathbf{v}_\theta)) = \mathbf{v}_{\theta-1}. \quad (12)$$

Denote $\mathbf{B} = (b_{i,j})_{i \in [n], j \in [m]}$, $\mathbf{x}_1 = (\mathbf{v}_1 \| \mathbf{w}_1)$ and $\mathbf{y}_1 = (\mathbf{w}_1 \| \mathbf{v}_1)$, and $\mathbf{u} = (u_1, \dots, u_n)^\top \in \mathbb{F}_2^n$. We have $\text{RE}(\mathbf{x}_1) = (f_1(\mathbf{x}_1), \dots, f_m(\mathbf{x}_1))^\top$ and $\text{RE}(\mathbf{y}_1) = (f_{m+1}(\mathbf{y}_1), \dots, f_{2m}(\mathbf{y}_1))^\top$ for some polynomials f_1, \dots, f_{2m} of degree c . Then Eq. (11) is equivalent to the following n degree- $(c+1)$ constraints:

$$\phi_i(\cdot) = \bar{j}_1 \cdot \left(\sum_{h=1}^m b_{i,h} f_h(\mathbf{x}_1) \right) + j_1 \cdot \left(\sum_{h=1}^m b_{i,h} f_{m+h}(\mathbf{y}_1) \right) - u_i, \quad \forall i \in [n]. \quad (13)$$

Here, the extra degree is due to multiplication with j_1 .

Similarly, Eq. (12) is equivalent to the following n degree- $(c+1)$ constraints:

$$\begin{aligned} \phi_{(\theta-1)n+i}(\cdot) &= \bar{j}_\theta \cdot \left(\sum_{h=1}^m b_{i,h} f_{2(\theta-1)m+h}(\mathbf{x}_\theta) \right) \\ &\quad + j_\theta \cdot \left(\sum_{h=1}^m b_{i,h} f_{2(\theta-1)m+m+h}(\mathbf{y}_\theta) \right) - v_{\theta,i}, \quad \forall i \in [n], \end{aligned} \quad (14)$$

where $\mathbf{x}_\theta = (\mathbf{v}_\theta \| \mathbf{w}_\theta)$, $\mathbf{y}_\theta = (\mathbf{w}_\theta \| \mathbf{v}_\theta)$, $\mathbf{v}_\theta = (v_{\theta,1}, \dots, v_{\theta,n})^\top$, and $f_{2\theta m-2m+1}, \dots, f_{2\theta m}$ are 2 m polynomials of degree c .

To this end, \mathcal{P} are prepared with ℓn polynomials $\phi_1(\cdot), \dots, \phi_{\ell n}(\cdot)$ of degree $c+1$, and possesses witness $\tilde{\mathbf{x}} = (j_1 \| \dots \| j_\ell \| \mathbf{v}_1 \| \mathbf{w}_1 \| \dots \| \mathbf{v}_\ell \| \mathbf{w}_\ell) \in \mathbb{F}_2^{\ell+2\ell n}$. Therefore, it can run the protocol $\Pi_{\text{dD-Rep}}^t$ and utilize the Fiat-Shamir transform to make it non-interactive. One can see that the witness size is optimal.

4.3 ZK of Plaintext Knowledge

Now, we provide a ZK of plaintext knowledge for the variant of randomized McEliece encryption schemes with regular noise described in Sect. 2.5. The prover needs to prove knowledge of a plaintext for a given ciphertext.

Specifically, the public inputs are $\mathbf{G} \in \mathbb{F}_2^{n_e \times k_e}$ and a ciphertext $\mathbf{c} \in \mathbb{F}_2^{n_e}$, and the secret inputs consist of $\mathbf{u} \in \mathbb{F}_2^{k_1}$, $\mathbf{m} \in \mathbb{F}_2^{k_2}$ as well as $\mathbf{e}' \in \mathbb{F}_2^k$ with $\frac{k}{c} \cdot 2^c = n_e$ such that

$$\mathbf{c} = \mathbf{G} \cdot \begin{pmatrix} \mathbf{u} \\ \mathbf{m} \end{pmatrix} \oplus \text{RE}(\mathbf{e}'). \quad (15)$$

Let $\mathbf{u} = (u_1, \dots, u_{k_1})^\top$, $\mathbf{m} = (m_{k_1+1}, \dots, m_{k_1+k_2})^\top$, $\mathbf{G} = (g_{i,j})_{i \in [n_e], j \in [k_e]}$, and $\mathbf{c} = (c_1, \dots, c_{n_e})^\top$. According to the technique in Sect. 3.2, we will have $\text{RE}(\mathbf{e}') = (f_1(\mathbf{e}'), \dots, f_{n_e}(\mathbf{e}'))^\top$ for some polynomials f_1, \dots, f_{n_e} of degree c . Then Eq. (15) is equivalent to the following n_e degree- c constraints:

$$\phi_i(\cdot) = \sum_{j=1}^{k_1} g_{i,j} \cdot u_j + \sum_{j=k_1+1}^{k_1+k_2} g_{i,j} \cdot m_j + f_i(\mathbf{e}') - c_i, \quad \forall i \in [n_e]. \quad (16)$$

To this end, \mathcal{P} prepares n_e public polynomials $\phi_1(\cdot), \dots, \phi_{n_e}(\cdot)$ of degree c , and the witness $\tilde{\mathbf{x}} = (\mathbf{u} \parallel \mathbf{m} \parallel \mathbf{e}') \in \mathbb{F}_2^{k_e+k}$. As a result, \mathcal{P} can run the protocol $\Pi_{\text{dD-Rep}}^t$ and make it non-interactive via the Fiat-Shamir transform.

5 ZK Protocols for Advanced Primitives

In this section, we provide new ZK protocols for code-based advanced privacy-preserving primitives, including ring signature scheme [69], a variant of group signature scheme [69], and fully dynamic attribute-based signature scheme [56]. Then we estimate the signature sizes of the above schemes by employing our ZK and Stern-like ZK [76]. The results show that the signature sizes utilizing our ZK protocols are two to three orders of magnitude smaller.

5.1 ZK for a Ring Signature Scheme

Being prepared with ZK protocols for proving the correctness of the regular encoding process from Sect. 3.2 and for proving the knowledge of an accumulated value from Sect. 4.2, we now provide a more efficient ZK protocol supporting the code-based ring signature scheme proposed by Nguyen et al. [69].

This protocol is an extension of the one from Sect. 4.2, where \mathcal{P} additionally convinces the verifier the following fact: He/She knows a secret key $\mathbf{x}_0, \mathbf{x}_1 \in \mathbb{F}_2^n$ such that

$$\mathbf{v}_\ell = \mathbf{B}_0 \cdot \text{RE}(\mathbf{x}_0) + \mathbf{B}_1 \cdot \text{RE}(\mathbf{x}_1). \quad (17)$$

In fact, we have already seen how to transform the above Eq. (17) into n constraints of degree c in Sect. 4.1. More specifically, there exist some polynomials $f_{2\ell m+1}, \dots, f_{2\ell m+m}$ of degree c such that

$$\text{RE}(\mathbf{x}_0 \parallel \mathbf{x}_1) = (f_{2\ell m+1}(\mathbf{x}_0 \parallel \mathbf{x}_1), \dots, f_{2\ell m+m}(\mathbf{x}_0 \parallel \mathbf{x}_1))^\top.$$

Therefore, Eq. (17) is equivalent to the following n degree- c relations:

$$\phi_{\ell n+i}(\cdot) = \sum_{h=1}^m b_{i,h} \cdot f_{2\ell m+h}(\mathbf{x}_0 \parallel \mathbf{x}_1) - v_{\ell,i}, \quad \forall i \in [n], \quad (18)$$

where $\mathbf{v}_\ell = (v_{\ell,1}, \dots, v_{\ell,n})^\top$ and $[\mathbf{B}_0 \parallel \mathbf{B}_1] = (b_{i,j})_{i \in [n], j \in [m]}$. At this point, \mathcal{P} has witness $\tilde{\mathbf{x}} = (j_1 \parallel \dots \parallel j_\ell \parallel \mathbf{v}_1 \parallel \mathbf{w}_1 \parallel \dots \parallel \mathbf{v}_\ell \parallel \mathbf{w}_\ell \parallel \mathbf{x}_0 \parallel \mathbf{x}_1) \in \mathbb{F}_2^{\ell+2\ell n+2n}$ and the newly appeared n degree- c constraints $\phi_{\ell n+1}(\cdot), \dots, \phi_{\ell n+n}(\cdot)$ in addition to the ℓn degree- $(c+1)$ constraints $\phi_1(\cdot), \dots, \phi_{\ell n}(\cdot)$ from Sect. 4.2. Therefore, it suffices for \mathcal{P} to run the protocol $\Pi_{\text{dD-Rep}}^t$ and then apply the Fiat-Shamir heuristic.

5.2 Parameters and Efficiency

We now estimate the concrete sizes of the above ring signature scheme using our ZK protocol and Stern-like protocol [69]. Note that the security of the scheme relies on the hardness of 2-RNSD $_{n,2n,c}$ problem. Also, we need to make sure the underlying proof systems achieve small enough soundness errors. Details of the parameters are given in Table 2.

Table 2. Parameters for the hash function $h_{\mathbf{B}}$, for the proof system from VOLEitH paradigm, and for the McEliece encryption cryptosystem that achieve 128-bit security and 256-bit security.

Parameters	Description	128-bit Security	256-bit Security
λ	Security level	128	256
n	Hash $h_{\mathbf{B}}$ output length	1280	2560
c	2-RNSD Parameter	8	8
$w = \frac{2n}{c}$	$h_{\mathbf{B}}$ input Hamming weight	320	640
$m = \frac{2n}{c} \cdot 2^c$	$h_{\mathbf{B}}$ input length	$5 \cdot 2^{14}$	$5 \cdot 2^{15}$
p	Base field \mathbb{F}_p	2	2
q	Extension field \mathbb{F}_q	2^8	2^8
τ	Repetition for VOLEitH	16	32
$s = \lambda + 16$	Universal hash parameter	144	272
$h = \lambda + 16$	Universal hash parameter	144	272
κ	Repetition for Stern	219	438
n_e	McEliece parameter	4096	8192
k_e	McEliece parameter	3328	6528
t_e	McEliece parameter	64	128

On the parameters of 2-RNSD problem. As discussed in Sect. 2.2, we choose parameters according to [3]. In particular, they chose $n = 1024, w = 128, m = 2^{21}$ and $n = 1984, w = 248, m = 31 \cdot 2^{16}$ for 128-bit and 256-bit security levels, respectively. However, we work in a setting where n, c uniquely determine w and m . Therefore, we adjust the parameters slightly by setting $c = 8$ and increasing n from 1024 to 1280 and from 1984 to 2560, respectively.

On the Parameters of the VOLEitH Proof System. We choose parameters according to the specification given in [7, 8], for 128-bit security. Regarding the parameters for 256-bit security level, we double the repetition parameter τ .

Repetition for Stern-like Protocols. The underlying ZK protocol for the above ring signature schemes used by Nguyen et al. [69] is Stern-like protocol [76]. Originally, it was designed to prove knowledge of a vector with exact Hamming weight. Later, it was adapted to prove various lattice-based and code-based linear and quadratic relations, e.g. [53, 57, 69], giving rise to various applications such as

ring signatures [54, 69], group signatures [53, 69], attribute-based signatures [56], group encryption [68] and so on. However, it has the main disadvantage of large soundness error $2/3$. Therefore, to achieve 2^{-128} and 2^{-256} soundness errors, one needs to repeat the protocol for 219 and 438 times.

Table 3. Ring signature sizes by employing our ZK proof system and Stern-like ZK arguments.

Ring size	128-bit security		256-bit security	
	This paper (KB)	Stern-type [69] (MB)	This paper (KB)	Stern-type [69] (MB)
2^5	35.12	32.26	140.24	129.04
2^7	45.12	43.93	180.25	175.74
2^{10}	60.13	61.44	240.26	245.78
2^{15}	85.14	90.63	340.28	362.51
2^{20}	110.15	119.81	440.30	479.25
2^{30}	160.17	178.18	640.34	712.72

Given the above parameters, we then give a detailed comparison about signature sizes for the ring signature scheme that employs our ZK protocol presented in Sect. 5.1 and that employs Stern-like protocols. Theoretically, both signature sizes are logarithmic in the size of the ring. Concretely, the performance of our ZK protocol appears to be significantly better. In particular, Table 3 shows that for 128-bit and 256-bit security levels, the signature sizes of [69] are around $934\times \sim 1140\times$ larger than ours for different ring sizes.

There are several reasons for the huge differences. One reason is that one needs to repeat Stern-type protocol for 219 times and 438 times to achieve negligible soundness error, while we only need to repeat VOLEitH proofs 16 and 32 times. Another reason is that their witness size is $(2 \cdot \ell + 1) \cdot (\frac{2n}{c} \cdot 2^c) + 2 \cdot \ell \cdot n$ bits while our witness size is just $\ell + 2\ell n + 2n$ bits.

Optimizations from [6]. As mentioned in Sect. 2.7, Baum et al. [6] proposed several optimizations to improve the realization of $\mathcal{F}_{\text{sVOLE}}^{p,q,S_{\Delta},C,l,\mathcal{L}}$. In particular, it brings the decommitment size, $\text{Cost}_{\text{decom}} = (2\lambda + r \cdot \lambda)\tau$ from (9), down to $2\lambda \cdot \tau + T_{\text{open}} \cdot \lambda$, where T_{open} is a threshold number considered in [6]. Let $T_{\text{open}} = 102$ and $T_{\text{open}} = 218$ for $\lambda = 128$ and $\lambda = 256$, then the decommitment sizes are reduced by around 416 bytes and 1212 bytes, respectively. Therefore, the figures of our constructions in Table 3 could be further improved. We, however, have to admit that these improvements are relatively small for privacy-preserving protocols, and will no longer consider them in GS and FDABS schemes.

5.3 ZK for a Group Signature Scheme

Next, we provide a more efficient ZK protocol supporting the code-based group signature scheme proposed by Nguyen et al. [69], with the modification that the McEliece scheme is replaced with one with regular noise.

This protocol is extended from the one in Sect. 5.1, for which an encryption layer is added. Specifically, \mathcal{P} additionally proves the following statement: He/She knows extra secret values $\mathbf{r}_1, \mathbf{r}_2 \in \mathbb{F}_2^{k_e - \ell}$, $\mathbf{e}'_1, \mathbf{e}'_2 \in \mathbb{F}_2^k$ with $\frac{k}{c} \cdot 2^c = n_e$ such that

$$\mathbf{c}_1 = \mathbf{G}_1 \cdot \begin{pmatrix} \mathbf{r}_1 \\ \text{bin}(j) \end{pmatrix} \oplus \text{RE}(\mathbf{e}'_1), \quad \text{and} \quad \mathbf{c}_2 = \mathbf{G}_2 \cdot \begin{pmatrix} \mathbf{r}_2 \\ \text{bin}(j) \end{pmatrix} \oplus \text{RE}(\mathbf{e}'_2), \quad (19)$$

where $\mathbf{G}_1, \mathbf{G}_2 \in \mathbb{F}_2^{n_e \times k_e}$ and $\mathbf{c}_1, \mathbf{c}_2 \in \mathbb{F}_2^{n_e}$.

We have described in Sect. 4.3 on transforming the above Eqs. (19) into polynomial constraints. For $\theta \in \{1, 2\}$, let

$$\begin{aligned} \mathbf{c}_\theta &= (c_{\theta,i})_{i \in [n_e]}, \quad \mathbf{G}_\theta = (g_{i,j}^{(\theta)})_{i \in [n_e], j \in [k_e]}, \quad \mathbf{r}_\theta = (r_{\theta,1}, \dots, r_{\theta,k_e - \ell})^\top, \\ \text{RE}(\mathbf{e}'_\theta) &= (f_{2\ell m + m + n_e(\theta-1)+1}(\mathbf{e}'_\theta), \dots, f_{2\ell m + m + n_e(\theta-1)+n_e}(\mathbf{e}'_\theta))^\top, \end{aligned}$$

for some polynomials $f_{2\ell m + m + 1}, \dots, f_{2\ell m + m + 2n_e}$ of degree c . Therefore, Eq. (19) is equivalent to the following $2n_e$ degree- c relations:

$$\begin{aligned} \phi_{\ell n + n + i}(\cdot) &= \sum_{h=1}^{k_e - \ell} g_{i,h}^{(1)} \cdot r_{1,j} + \sum_{h=1}^{\ell} g_{i,k_e - \ell + h}^{(1)} \cdot j_h \\ &\quad + f_{2\ell m + m + i}(\mathbf{e}'_1) - c_{1,i}, \quad \forall i \in [n_e], \\ \phi_{\ell n + n + n_e + i}(\cdot) &= \sum_{h=1}^{k_e - \ell} g_{i,h}^{(2)} \cdot r_{2,j} + \sum_{h=1}^{\ell} g_{i,k_e - \ell + h}^{(2)} \cdot j_h \\ &\quad + f_{2\ell m + m + n_e + i}(\mathbf{e}'_2) - c_{2,i}, \quad \forall i \in [n_e]. \end{aligned}$$

Now, \mathcal{P} has witness $\tilde{\mathbf{x}} \in \mathbb{F}_2^{\ell + 2\ell n + 2n + 2(k_e - \ell) + 2k}$ of the following form

$$\tilde{\mathbf{x}} = (j_1 \parallel \dots \parallel j_\ell \parallel \mathbf{v}_1 \parallel \mathbf{w}_1 \parallel \dots \parallel \mathbf{v}_\ell \parallel \mathbf{w}_\ell \parallel \mathbf{x}_0 \parallel \mathbf{x}_1 \parallel \mathbf{r}_1 \parallel \mathbf{r}_2 \parallel \mathbf{e}'_1 \parallel \mathbf{e}'_2) \quad (20)$$

and the newly appeared $2n_e$ degree- c constraints $\phi_{\ell n + n + 1}(\cdot), \dots, \phi_{\ell n + n + 2n_e}(\cdot)$ in addition to the $\ell n + n$ constraints $\phi_1(\cdot), \dots, \phi_{\ell n + n}(\cdot)$ from Sect. 5.1. Now \mathcal{P} can proceed as before by running the protocol $\Pi_{\text{dD-Rep}}^t$ and then applying the Fiat-Shamir heuristic.

5.4 Parameters and Efficiency

We now estimate the concrete sizes of group signature scheme using our ZK protocol and Stern-like protocol [69]. The security of the scheme relies on the hardness of 2-RNSD $_{n,2n,c}$ problem, on the CPA-security of the McEliece encryption scheme, as well as the security of the supporting ZK protocols. We use the same parameters proposed in Table 2.

On the Parameters of McEliece Encryption Scheme. We choose parameters for McEliece cryptosystem following the document [2] with minor adaptations. Since we modify the noise vector to be a regular vector, the CPA-security of the McEliece encryption scheme now depends on the decisional RSD problem as discussed in Sect. 2.5. However, this is not an issue as shown in [59, Table 1, Table 2], the usage of regular noise for LPN and SD problems does not reduce the bit security significantly. In fact, one can always choose slightly larger parameters to obtain targeted security levels. In our setting, we then slightly increase the Goppa code length n_e (and hence the dimension k_e) so that it is the form of $\frac{k}{c} \cdot 2^c$ with $\frac{k}{c} = t_e$.

Table 4. Group signature sizes by employing our ZK proof system and Stern-like ZK arguments.

Group size	128-bit security		256-bit security	
	This paper (KB)	Stern-type [69] (MB)	This paper (KB)	Stern-type [69] (MB)
2^5	49.60	33.27	197.19	133.02
2^7	59.60	44.94	237.19	179.72
2^{10}	74.59	62.45	297.18	249.76
2^{15}	99.58	91.63	397.16	366.50
2^{20}	124.57	120.82	497.14	483.23
2^{30}	174.55	179.18	697.10	716.70

With the above parameters, we then estimate signature sizes using our ZK and Stern-like ZK for various group sizes. Details are in Table 4. The results also show the superiority of our ZK protocols. In particular, for 128-bit and 256-bit security, the signature sizes of [69] are around $683\times \sim 1053\times$ larger than ours for different group sizes.

5.5 ZK for a Fully Dynamic Attribute-Based Signature Scheme

Quite recently, Nguyen et al. [56] proposed a fully dynamic attribute-based signature (FDABS) scheme from codes. The scheme employs a refined Stern-like protocol and is proven secure in the quantum oracle model (QROM) using the variant of Unruh transform [77] presented in [43]. To the best of our knowledge, we are unaware of existing works on making VOLEitH protocol secure in QROM. A related work by Aguilar-Melchor et al. [65] presented a security proof for Hypercube-SDitH [64] in the QROM. It remains open if one can apply their techniques to the VOLEitH paradigm. Therefore, we provide a new ZK for their FDABS scheme that is only secure in the ROM and then compare efficiency with their degraded variant.

We now provide our new ZK protocol. It is an extension of the one from Sect. 4.2, where \mathcal{P} additionally convinces the verifier the following facts: He/she knows an attribute $\mathbf{x} \in \{0, 1\}^L$ together with a randomness $\mathbf{r} \in \{0, 1\}^k$ such that

$$\mathbf{v}_\ell = \mathbf{C}_0 \cdot \text{RE}(\mathbf{x}) \oplus \mathbf{C}_1 \cdot \text{RE}(\mathbf{r}); \quad (21)$$

$$\text{wt}(\mathbf{v}_\ell) = 1 \pmod{2}; \quad (22)$$

$$P(\mathbf{x}) = 1, \quad (23)$$

where P is an arbitrary binary circuit with L bit inputs and K multiplication gates.

We have already shown how to transform the Eq. (21) into polynomial constraints. Recall that $\mathbf{C}_0 = (c_{i,j})_{i \in [n], j \in [m_0]}$, $\mathbf{C}_1 = (c_{i,m_0+j})_{i \in [n], j \in [m_1]}$ with $m_0 = \frac{L}{c} \cdot 2^c$ and $m_1 = \frac{k}{c} \cdot 2^c$. Let $\mathbf{v}_\ell = (v_{\ell,1}, \dots, v_{\ell,n})^\top$. Then Eq. (21) is equivalent to the following n degree- c relations

$$\phi_{\ell n+i}(\cdot) = \sum_{h=1}^{m_0+m_1} c_{i,h} f_{2\ell m+h}(\mathbf{x} \parallel \mathbf{r}) - v_{\ell,i}, \quad \forall i \in [n].$$

Regarding (22), it asks to prove that the Hamming weight of \mathbf{v}_ℓ is odd. We then observe that this is equivalent to proving

$$v_{\ell,1} + v_{\ell,2} + \dots + v_{\ell,n} = 1.$$

Define $\phi_{\ell n+n+1}(X_1, \dots, X_n) = X_1 + \dots + X_n - 1 \in \mathbb{F}_2[X_1, \dots, X_n]$. Then equation (22) is further equivalent to the following linear polynomial

$$\phi_{\ell n+n+1}(\cdot) = v_{\ell,1} + v_{\ell,2} + \dots + v_{\ell,n} - 1. \quad (24)$$

In terms of (23), as observed by Ling et al. [56], it is equivalent to the following K quadratic equations:

$$\begin{cases} \phi_{\ell n+n+1+1}(\cdot) = x_{\alpha(1)} \cdot x_{\beta(1)} \oplus x_{L+1} - 1, \\ \dots \\ \phi_{\ell n+n+1+K-1}(\cdot) = x_{\alpha(K-1)} \cdot x_{\beta(K-1)} \oplus x_{L+K-1} - 1, \\ \phi_{\ell n+n+1+K}(\cdot) = x_{\alpha(K)} \cdot x_{\beta(K)} \oplus x_{L+K} - 1, \end{cases} \quad (25)$$

where x_{L+1}, \dots, x_{L+K} are the output wire values of multiplication gates and $\alpha, \beta: \{1, \dots, K\} \rightarrow \{1, \dots, L+K-1\}$ are two functions specifying the topology of the circuit P .

Now \mathcal{P} has witness $\tilde{\mathbf{x}} \in \mathbb{F}_2^{\ell+2\ell n+L+k+K}$ of the following form

$$\tilde{\mathbf{x}} = (j_1 \parallel \dots \parallel j_\ell \parallel \mathbf{v}_1 \parallel \mathbf{w}_1 \parallel \dots \parallel \mathbf{v}_\ell \parallel \mathbf{w}_\ell \parallel \mathbf{x} \parallel \mathbf{r} \parallel x_{L+1} \parallel \dots \parallel x_{L+K}), \quad (26)$$

and the newly appeared $n+1+K$ constraints $\phi_{\ell n+1}(\cdot), \dots, \phi_{\ell n+n+1+K}(\cdot)$ in addition to the ℓn constraints $\phi_1(\cdot), \dots, \phi_{\ell n}(\cdot)$ from Sect. 4.2. Now \mathcal{P} can proceed as before by running the protocol $\Pi_{\text{dD-Rep}}^t$ and then applying the Fiat-Shamir heuristic.

We remark that our technique of handling odd Hamming weight vectors can be employed to upgrade the static GS scheme [13] to a fully dynamic one with the same signature sizes. Specifically, we first modify the fully dynamic GS scheme [79] by restricting the user public key \mathbf{v}_ℓ^3 to have odd Hamming weight. Then we modify the ZK presented in Sect. 5.3 to additionally show that \mathbf{v}_ℓ has odd Hamming weight. Since this change does not increase the witness length, the signature sizes remain the same.

5.6 Parameters and Efficiency

We now estimate the concrete sizes of FDABS scheme using our ZK protocol and Stern-like protocol [56]. The security of the scheme relies on the hardness of $2\text{-RNSD}_{n,2n,c}$ and $2\text{-RNSD}_{n,L+k,c}$ problems, as well as the security of the supporting ZK protocols. To this end, we use the same parameters proposed in Table 2 and let the bit length of attribute be $L = 128$ and the bit length of randomnesses for committing the attributes be $k = n + 2\lambda$.

We calculate various signature sizes regarding different security parameters and (ℓ, K) pairs, where 2^ℓ is the maximum number of attributes allowed in the system and K is the number of multiplication gates representing the policy P . Details are in Table 5. The results further confirm the superiority of our ZK protocols. For $K = 2^9$, the signature sizes of [56] are $783\times \sim 839\times$ larger than ours. For $K = 2^{16}$, the differences of the two ZK are smaller, and their signature sizes are $288\times \sim 550\times$ larger than ours. The main reason is that their witness sizes are dominated by the term $2\ell \cdot \frac{2n}{c} \cdot 2^c$ and thus less sensitive to changes of the policy size K . In contrast, our witness size is $\ell + 2\ell n + L + k + K$, and thus is susceptible to the changes of K .

Table 5. Signature sizes of the FDABS scheme by employing our ZK proof system and Stern-like ZK arguments. The bit length of the attribute in the following instances are always chosen as $L = 128$.

$(2^\ell, K)$	128-bit security		256-bit security	
	This paper (KB)	Stern-type [56] (MB)	This paper (KB)	Stern-type [56] (MB)
$(2^{10}, 2^9)$	59.38	45.41	234.76	181.29
$(2^{10}, 2^{16})$	186.38	52.20	488.76	194.87
$(2^{15}, 2^9)$	84.39	67.30	334.78	268.85
$(2^{15}, 2^{16})$	211.39	74.08	588.78	282.43
$(2^{20}, 2^9)$	109.40	89.18	434.80	356.40
$(2^{20}, 2^{16})$	236.40	95.97	688.80	369.98

³ Originally, it is required to be non-zero.

6 A New Code-Based Signature Scheme

A standard paradigm to construct a signature scheme is to first design a public-coin ZK protocol for proving the knowledge of a preimage of a one-way function and then apply the Fiat-Shamir [44] transform to make it non-interactive. Therefore, the technique from Sect. 3.2 directly implies a code-based signature scheme, which we name ReSolveD+ and has slightly smaller signature sizes than the state-of-the-art code-based signature scheme ReSolveD [32] based on RSD problems.

6.1 Description of the Signature Scheme

The secret key is binary string $\mathbf{x} \in \mathbb{F}_2^n$, and the public key is $\mathbf{y} = \mathbf{B} \cdot \text{RE}(\mathbf{x})$, where $\mathbf{B} = (b_{i,j})_{i \in [n], j \in [m]} \in \mathbb{F}_2^{n \times m}$ for $c|n$ and $m = \frac{n}{c} \cdot 2^c$ are public parameters. To sign a message $M \in \{0, 1\}^*$, the signer proves knowledge of \mathbf{x} such that $\mathbf{y} = (y_1, \dots, y_n)^\top = \mathbf{B} \cdot \text{RE}(\mathbf{x})$. In particular, the signer prepares n polynomials of degree c :

$$g_i(\cdot) = \sum_{j=1}^m b_{i,j} f_j(X_1, \dots, X_n) - y_i, \quad \forall i \in [n],$$

and the witness $\mathbf{x} \in \mathbb{F}_2^n$. Next, it runs the protocol $\Pi_{\text{dD-Rep}}^t$ and makes it non-interactive via the Fiat-Shamir transform. Let the resultant proof be π , which would be the signature. Verification of the signature is to verify the proof π . Correctness and security directly follows from that of $\Pi_{\text{dD-Rep}}^t$ and the design paradigm, based on the hardness of the $\text{RSD}_{n,n,c}$ problem.

Table 6. Parameters for ReSolveD+ Signature Scheme when $c = \{2, 3, 4\}$.

Scheme	Parameter set					Estimated Bit Security
	n	c	$m = \frac{n}{c} \cdot 2^c$	$w = \frac{n}{c}$	τ	
ReSolveD + -128-Var1-2	8922	2	1784	446	14	128.10
ReSolveD + -128-Var2-2	8922	2	1784	446	10	128.10
ReSolveD + -128-Var1-3	4533	3	1208	151	14	128.31
ReSolveD + -128-Var2-3	4533	3	1208	151	10	128.31
ReSolveD + -128-Var1-4	3324	4	1328	83	14	128.33
ReSolveD + -128-Var2-4	3324	4	1328	83	10	128.33

6.2 Parameters and Efficiency

We follow the approach in [29] to select parameters n, c . In particular, we estimate the complexity of linearization attack, ISD attack, and birthday paradox

according to formulas [29], and take their minimum as the estimation of security level. Using this estimation, we choose the smallest parameter n by fixing $c = 2, 3, 4$, respectively so that it has complexity estimation 2^{128} following the footprint of [32]. Details are in Table 6. Regarding the parameters for VOLEitH, we employ the same optimizations adopted by Cui et al. [32] (instead of the non-optimized ones from Sect. 3.1), to have a fair comparison with them. In Table 7, we compare the signature sizes of our signature scheme ReSolveD+ with ReSolveD for different parameter sets. The results show that we achieve smallest signature sizes when $c = 3$. In addition, our scheme ReSolveD+ has slightly shorter signature sizes than [32] when $c = 3$ and $c = 4$. Note that Baum et al. [6] recently obtained better performances for FAEST signature scheme by proposing several optimizations. Since those optimizations apply to all protocols within the VOLEitH paradigm, both signature sizes of ReSolveD and ReSolveD+ could be further reduced by a few hundred bytes. In particular, let $T_{\text{open}} = 112$ for $\tau = 14$ and $T_{\text{open}} = 102$ for $\tau = 10$. Then the signature sizes are reduced by 256 bytes and 416 bytes, respectively. The optimized signature sizes are given in Table 7 in blue color.

Table 7. Comparison of our signature schemes for different choices of c, τ, T_{open} with the signature scheme proposed by Cui et al. [32] for the same security levels. The percentages in parenthesis are the increases/decreases of signature sizes compared to [32].

Scheme parameters		Signature sizes in bytes			
		CLY+24 [32]	$c = 2$	$c = 3$	$c = 4$
$\tau = 14$	$T_{\text{open}} = -$	4082	4572(+12.0%)	4026(-1.4%)	4040(-1.0%)
	$T_{\text{open}} = 112$	3826	4316(+12.9%)	3770(-1.5%)	3784(-1.1%)
$\tau = 10$	$T_{\text{open}} = -$	3510	3860(+10.0%)	3470(-1.1%)	3480(-0.9%)
	$T_{\text{open}} = 102$	3094	3444(+11.3%)	3054(-1.3%)	3064(-1.0%)

We also compare our signature scheme with some other post-quantum signature schemes, for 128-bit security level in Table 8. The results show that our signature sizes are competitive with those schemes and are smallest among schemes based on SD and regular SD problems. Also, Bidoux et al. [18] proposed signature schemes based on Rank SD and MinRank problems from MPCitH and VOLEitH paradigm, we only include the variants that employ the optimizations from [6] within the VOLEitH paradigm. Adj et al. [1] proposed a signature scheme based on MinRank problem preceding [18] and had slightly larger signature sizes. Therefore, we do not include their results [1] in the table.

Conclusions and Open Questions. In this work, we advanced the state-of-the-art code-based cryptography by proposing new ZK protocols from VOLEitH paradigm. In particular, we presented ZK protocols for proving the correctness of the regular encoding process and various code-based relations. Built upon these

Table 8. Comparison of our scheme with some post-quantum signature schemes, targeting 128-bit security level. All the signature schemes within the VOLEitH paradigm are optimized using the techniques from [6].

Scheme	Sizes in KB			Assumptions	Paradigm
	sig	pk	sig + pk		
BGKM23 [19]-Sig1	24.0	0.1	24.1	SD over \mathbb{F}_2	Stern-type
BGKM23 [19]-Sig2	19.3	0.2	19.5	(QC)SD over \mathbb{F}_2	
BGKM23 [19]-Sig3	15.6	0.2	15.8	(QC)SD over \mathbb{F}_2	
FJR22 [40]-Var2s	11.8	0.09	11.89	SD over \mathbb{F}_2	MPCitH
FJR22 [40]-Var3f	11.5	0.14	11.64		
FJR22 [40]-Var3s	8.26	0.14	8.4		
CCJ23 [29]-rsd-f	12.52	0.09	12.61	RSD over \mathbb{F}_2	MPCitH
CCJ23 [29]-rsd-m1	9.69	0.09	9.78		
CCJ23 [29]-rsd-m2	9.13	0.09	9.22		
CCJ23 [29]-rsd-s	8.55	0.09	8.64		
MGH+23 [64]-faster	11.83	0.14	11.97	SD over \mathbb{F}_{256}	MPCitH
MGH+23 [64]-short	8.28	0.14	8.42		
MGH+23 [64]-shorter	6.63	0.14	6.77		
MGH+23 [64]-shortest	5.56	0.14	5.7		
[65]-Vanilla-short	8.27	0.14	8.6	SD over \mathbb{F}_{256}	MPCitH
[65]-Vanilla-shorter	6.6	0.14	6.94		
[65]-Pow-short	7.78	0.14	8.11		(QROM)
[65]-Pow-shorter	6.06	0.14	6.34		
BCC+24 [26]-fast	7.07	0.10	7.17	RSD over \mathbb{F}_2	MPCitH
BCC+24 [26]-medium	5.73	0.10	5.83		
BCC+24 [26]-compact	5.13	0.10	5.23		
[6] FAESTER-128 s	4.49	0.03	4.52	OW of AES128	VOLEitH
[6] FAESTER-128f	5.91	0.03	5.94		
[6] FAESTER-EM-128 s	4.07	0.03	4.10		
[6] FAESTER-EM-128f	5.32	0.03	5.35		
[6] FAESTER-d7-128 s	4.27	0.03	4.30		
[6] FAESTER-d7-128f	5.60	0.03	5.63		
[18] RSD _s -short	3.51	0.05	3.56	Rank SD	VOLEitH
[18] RSD _s -shortest	2.84	0.05	2.89		
[18] MinRank-short	3.46	0.03	3.49	MinRank	VOLEitH
[18] MinRank-shortest	2.81	0.03	2.84		
[32] ReSolveD-128-Var1	3.74	0.08	3.82	RSD over \mathbb{F}_2	VOLEitH
[32] ReSolveD-128-Var2	3.02	0.08	3.10		
ReSolveD + -128-Var1-3	3.68	0.06	3.74	RSD over \mathbb{F}_2	VOLEitH
ReSolveD + -128-Var2-3	2.98	0.06	3.04		

ZK protocols, we obtained privacy-preserving signatures whose signature sizes are significantly smaller than those based on Stern-like ZK protocols. We view the problem of improving the computational efficiency, particularly decreasing the number of multiplications over large finite fields required in $\Pi_{\text{dD-Rep}}^t$ and improving the realization of $\mathcal{F}_{\text{SVOLE}}^{p,q,S_\Delta,C,l,\mathcal{L}}$, as fascinating opening questions for future investigations.

Acknowledgements. We would like to thank Liping Wang, Khoa Nguyen, Hongrui Cui, Hanlin Liu, Xindong Liu, Yizhou Yao, and Hongqing Liu for their valuable discussions of this work. We are also very grateful for the insightful comments and suggestions from the anonymous reviewers of ASIACRYPT 2024. This work was supported in part by the National Key Research and Development Program under Grants 2020YFA0712300 and 2022YFA1004900, and the National Natural Science Foundation of China under Grant numbers 62272303 and 12101404.

References

1. G. Adj, L. Rivera-Zamarripa, and J. A. Verbel. Minrank in the head - short signatures from zero-knowledge proofs. In *AFRICACRYPT 2023*, volume 14064 of *LNCS*, pages 3–27. Springer, 2023.
2. M. R. Albrecht, D. J. Bernstein, T. Chou, C. Cid, J. Gilcher, T. Lange, V. Maram, I. Von Maurich, R. Misoczki, R. Niederhagen, et al. Classic mceliece: conservative code-based cryptography. 2022. <https://classic.mceliece.org/nist.html>.
3. D. Augot, M. Finiasz, P. Gaborit, S. Manuel, and N. Sendrier. Sha-3 proposal: Fsb. *Submission to NIST*, pages 81–85, 2008. <https://www.rocq.inria.fr/secret/CBCrypto/fsbdoc.pdf>.
4. D. Augot, M. Finiasz, and N. Sendrier. A fast provably secure cryptographic hash function. *IACR Cryptol. ePrint Arch.*, page 230, 2003.
5. D. Augot, M. Finiasz, and N. Sendrier. A family of fast syndrome based cryptographic hash functions. In *Mycrypt 2005*, volume 3715 of *LNCS*, pages 64–83. Springer, 2005.
6. C. Baum, W. Beullens, S. Mukherjee, E. Orsini, S. Ramacher, C. Rechberger, L. Roy, and P. Scholl. One tree to rule them all: Optimizing GGM trees and owfs for post-quantum signatures. *IACR Cryptol. ePrint Arch.*, page 490, 2024.
7. C. Baum, L. Braun, C. D. de Saint Guilhem, M. Kloof, C. Majenz, S. Mukherjee, S. Ramacher, C. Rechberger, E. Orsini, L. Roy, et al. Faest: Algorithm specifications. 2023.
8. C. Baum, L. Braun, C. D. de Saint Guilhem, M. Kloof, E. Orsini, L. Roy, and P. Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from vole-in-the-head. In *CRYPTO 2023*, volume 14085 of *LNCS*, pages 581–615. Springer, 2023.
9. C. Baum, L. Braun, A. Munch-Hansen, and P. Scholl. Moz \mathbb{Z}_{2^k} arella: Efficient vector-ole and zero-knowledge proofs over \mathbb{Z}_{2^k} . In *CRYPTO 2022*, volume 13510 of *LNCS*, pages 329–358. Springer, 2022.
10. C. Baum, S. Dittmer, P. Scholl, and X. Wang. Sok: vector ole-based zero-knowledge protocols. *Des. Codes Cryptogr.*, 91(11):3527–3561, 2023.
11. C. Baum, A. J. Malozemoff, M. B. Rosen, and P. Scholl. Mac’n’cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In *CRYPTO 2021*, volume 12828 of *LNCS*, pages 92–122. Springer, 2021.

12. M. Bellare and G. Fuchsbauer. Policy-based signatures. In *PKC 2014*, volume 8383 of *LNCS*, pages 520–537. Springer, 2014.
13. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, 2003.
14. J. C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital signatures (extended abstract). In *EUROCRYPT 1993*, volume 765 of *LNCS*, pages 274–285. Springer, 1993.
15. D. J. Bernstein, T. Lange, R. Niederhagen, C. Peters, and P. Schwabe. Fsbday: Implementing wagner’s generalized birthday attack against the sha-3 round-1 candidate fsb. In *INDOCRYPT 2009*, volume 5922 of *LNCS*, pages 18–38. Springer, 2009.
16. D. J. Bernstein, T. Lange, C. Peters, and P. Schwabe. Faster 2-regular information-set decoding. In *IWCC 2011*, volume 6639 of *LNCS*, pages 81–98. Springer, 2011.
17. D. J. Bernstein, T. Lange, C. Peters, and P. Schwabe. Really fast syndrome-based hashing. In *AFRICACRYPT 2011*, volume 6737 of *LNCS*, pages 134–152. Springer, 2011.
18. L. Bidoux, T. Feneuil, P. Gaborit, R. Neveu, and M. Rivain. Dual support decomposition in the head: Shorter signatures from rank SD and minrank. *IACR Cryptol. ePrint Arch.*, page 541, 2024.
19. L. Bidoux, P. Gaborit, M. Kulkarni, and V. Mateu. Code-based signatures from new proofs of knowledge for the syndrome decoding problem. *Des. Codes Cryptogr.*, 91(2):497–544, 2023.
20. D. Boneh, S. Eskandarian, and B. Fisch. Post-quantum EPID signatures from symmetric primitives. In M. Matsui, editor, *CT-RSA 2019*, volume 11405 of *LNCS*, pages 251–271. Springer, 2019.
21. E. Boyle, G. Couteau, N. Gilboa, and Y. Ishai. Compressing vector OLE. In *CCS 2018*, pages 896–912. ACM, 2018.
22. 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*, pages 291–308. ACM, 2019.
23. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Efficient pseudo-random correlation generators: Silent OT extension and more. In *CRYPTO 2019*, volume 11694 of *LNCS*, pages 489–518. Springer, 2019.
24. P. Briaud and M. Øygarden. A new algebraic approach to the regular syndrome decoding problem and implications for PCG constructions. In *EUROCRYPT 2023*, volume 14008 of *LNCS*, pages 391–422. Springer, 2023.
25. E. F. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *CCS 2004*, pages 132–145. ACM, 2004.
26. D. Bui, E. Carozza, G. Couteau, D. Goudarzi, and A. Joux. Short signatures from regular syndrome decoding, revisited. *IACR Cryptol. ePrint Arch.*, page 252, 2024.
27. J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology - CRYPTO 2002*, volume 2442 of *LNCS*, pages 61–76. Springer, 2002.
28. C. Carlet, editor. *Boolean Functions for Cryptography and Coding Theory*. Cambridge University Press, Cambridge, 2020.
29. E. Carozza, G. Couteau, and A. Joux. Short signatures from regular syndrome decoding in the head. In *EUROCRYPT 2023*, volume 14008 of *LNCS*, pages 532–563. Springer, 2023.

30. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.
31. D. Chaum and E. van Heyst. Group signatures. In *Advances in Cryptology - EURO-CRYPT 1991*, volume 547 of *LNCS*, pages 257–265. Springer, 1991.
32. H. Cui, H. Liu, D. Yan, K. Yang, Y. Yu, and K. Zhang. Resolved: Shorter signatures from regular syndrome decoding and vole-in-the-head. In *PKC 2024*, volume 14601 of *LNCS*, pages 229–258. Springer, 2024.
33. D. Derler, S. Ramacher, and D. Slamanig. Post-quantum zero-knowledge proofs for accumulators with applications to ring signatures from symmetric-key primitives. In *PQCrypto 2018*, volume 10786 of *LNCS*, pages 419–440. Springer, 2018.
34. S. Dittmer, Y. Ishai, S. Lu, and R. Ostrovsky. Improving line-point zero knowledge: Two multiplications for the price of one. In *CCS 2022*, pages 829–841. ACM, 2022.
35. S. Dittmer, Y. Ishai, and R. Ostrovsky. Line-point zero knowledge and its applications. In *ITC 2021*, volume 199 of *LIPICs*, pages 5:1–5:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.
36. A. El Kaafarani and S. Katsumata. Attribute-based signatures for unbounded circuits in the ROM and efficient instantiations from lattices. In *PKC 2018*, volume 10770 of *LNCS*, pages 89–119. Springer, 2018.
37. A. Esser, R. Kübler, and A. May. LPN decoded. In *CRYPTO 2017*, volume 10402 of *LNCS*, pages 486–514. Springer, 2017.
38. A. Esser and P. Santini. Not just regular decoding: Asymptotics and improvements of regular syndrome decoding attacks. *IACR Cryptol. ePrint Arch.*, page 1568, 2023.
39. M. F. Ezerman, H. T. Lee, S. Ling, K. Nguyen, and H. Wang. Provably secure group signature schemes from code-based assumptions. *IEEE Trans. Inf. Theory*, 66(9):5754–5773, 2020.
40. T. Feneuil, A. Joux, and M. Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In *CRYPTO 2022*, volume 13508 of *LNCS*, pages 541–572. Springer, 2022.
41. T. Feneuil, A. Joux, and M. Rivain. Shared permutation for syndrome decoding: new zero-knowledge protocol and code-based signature. *Des. Codes Cryptogr.*, 91(2):563–608, 2023.
42. T. Feneuil and M. Rivain. Threshold linear secret sharing to the rescue of mpc-in-the-head. In *ASIACRYPT 2023*, volume 14438 of *LNCS*, pages 441–473. Springer, 2023.
43. H. Feng, J. Liu, and Q. Wu. Secure stern signatures in quantum random oracle model. In *Information Security - ISC 2019*, volume 11723 of *LNCS*, pages 425–444. Springer, 2019.
44. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO 1986*, volume 263 of *LNCS*, pages 186–194. Springer, 1986.
45. C. Ganesh, C. Orlandi, M. Pancholi, A. Takahashi, and D. Tschudi. Fiat-shamir bulletproofs are non-malleable (in the random oracle model). *IACR Cryptol. ePrint Arch.*, page 147, 2023.
46. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.
47. S. Gueron, E. Persichetti, and P. Santini. Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup. *Cryptogr.*, 6(1):5, 2022.

48. C. Hazay, E. Orsini, P. Scholl, and E. Soria-Vazquez. Tinykeys: A new approach to efficient multi-party computation. In *CRYPTO 2018*, volume 10993 of *LNCS*, pages 3–33. Springer, 2018.
49. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM J. Comput.*, 39(3):1121–1152, 2009.
50. C. Jeudy, A. Roux-Langlois, and O. Sanders. Lattice signature with efficient protocols, application to anonymous credentials. In *CRYPTO 2023*, volume 14082 of *LNCS*, pages 351–383. Springer, 2023.
51. J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *CCS 2018*, pages 525–537. ACM, 2018.
52. A. Kiayias, Y. Tsiounis, and M. Yung. Group encryption. In *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 181–199. Springer, 2007.
53. B. Libert, S. Ling, F. Mouhartem, K. Nguyen, and H. Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In *ASIACRYPT 2016*, volume 10032 of *LNCS*, pages 373–403, 2016.
54. B. Libert, S. Ling, K. Nguyen, and H. Wang. Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In *EUROCRYPT 2016*, volume 9666 of *LNCS*, pages 1–31. Springer, 2016.
55. F. Lin, C. Xing, and Y. Yao. More efficient zero-knowledge protocols over \mathbb{Z}_{2^k} via galois rings. *IACR Cryptol. ePrint Arch.*, page 150, 2023.
56. S. Ling, K. Nguyen, D. H. Phan, K. H. Tang, H. Wang, and Y. Xu. Fully dynamic attribute-based signatures for circuits from codes. In *PKC 2024*, volume 14601 of *LNCS*, pages 37–73. Springer, 2024.
57. S. Ling, K. Nguyen, D. Stehlé, and H. Wang. Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In *PKC 2013*, volume 7778 of *LNCS*, pages 107–124. Springer, 2013.
58. H. Liu, X. Wang, K. Yang, and Y. Yu. The hardness of LPN over any integer ring and field for PCG applications. *IACR Cryptol. ePrint Arch.*, page 712, 2022.
59. H. Liu, X. Wang, K. Yang, and Y. Yu. The hardness of LPN over any integer ring and field for PCG applications. In *EUROCRYPT 2024*, volume 14656 of *LNCS*, pages 149–179. Springer, 2024.
60. X. Liu and L. Wang. Short code-based one-out-of-many proofs and applications. In *PKC 2024*, volume 14602 of *LNCS*, pages 370–399. Springer, 2024.
61. V. Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 598–616. Springer, 2009.
62. V. Lyubashevsky and N. K. Nguyen. BLOOM: bimodal lattice one-out-of-many proofs and applications. In S. Agrawal and D. Lin, editors, *ASIACRYPT 2022*, volume 13794 of *LNCS*, pages 95–125. Springer, 2022.
63. R. J. McEliece. A public-key cryptosystem based on algebraic. *Coding Thv*, 4244:114–116, 1978.
64. C. A. Melchor, N. Gama, J. Howe, A. Hülsing, D. Joseph, and D. Yue. The return of the sdith. In *EUROCRYPT 2023*, volume 14008 of *LNCS*, pages 564–596. Springer, 2023.
65. C. A. Melchor, A. Hülsing, D. Joseph, C. Majenz, E. Ronen, and D. Yue. Sdith in the QROM. In *ASIACRYPT 2023*, volume 14444 of *LNCS*, pages 317–350. Springer, 2023.
66. R. C. Merkle. A certified digital signature. In *CRYPTO 1989*, volume 435 of *LNCS*, pages 218–238. Springer, 1989.

67. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC 1990*, pages 427–437. ACM, 1990.
68. K. Nguyen, R. Safavi-Naini, W. Susilo, H. Wang, Y. Xu, and N. Zeng. Group encryption: Full dynamicity, message filtering and code-based instantiation. In *PKC 2021*, volume 12711 of *LNCS*, pages 678–708. Springer, 2021. Full version is available at <https://eprint.iacr.org/2021/226>.
69. K. Nguyen, H. Tang, H. Wang, and N. Zeng. New code-based privacy-preserving cryptographic constructions. In *ASIACRYPT 2019*, volume 11922 of *LNCS*, pages 25–55. Springer, 2019.
70. R. Nojima, H. Imai, K. Kobara, and K. Morozov. Semantic security for the mceliece cryptosystem without random oracles. *Des. Codes Cryptogr.*, 49(1-3):289–305, 2008.
71. R. O’Donnell. *Analysis of Boolean Functions*. Cambridge University Press, 2014.
72. E. Prange. The use of information sets in decoding cyclic codes. *IRE Trans. Inf. Theory*, 8(5):5–9, 1962.
73. R. L. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. In *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, 2001.
74. L. Roy. Softspokenot: Quieter OT extension from small-field silent VOLE in the minicrypt model. In *CRYPTO 2022*, volume 13507 of *LNCS*, pages 657–687. Springer, 2022.
75. A. Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS 1999*, pages 543–553. IEEE Computer Society, 1999.
76. J. Stern. A new paradigm for public key identification. *IEEE Trans. Inf. Theory*, 42(6):1757–1768, 1996.
77. D. Unruh. Non-interactive zero-knowledge proofs in the quantum random oracle model. In *EUROCRYPT 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 755–784. Springer, 2015.
78. D. A. Wagner. A generalized birthday problem. In *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, 2002.
79. L. Wang, J. Chen, H. Dai, and C. Tao. Efficient code-based fully dynamic group signature scheme. *Theor. Comput. Sci.*, 990:114407, 2024.
80. C. Weng, K. Yang, J. Katz, and X. Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *IEEE Symposium on Security and Privacy 2021*, pages 1074–1091. IEEE, 2021.
81. C. Weng, K. Yang, Z. Yang, X. Xie, and X. Wang. Antman: Interactive zero-knowledge proofs with sublinear communication. In *CCS 2022*, pages 2901–2914. ACM, 2022.
82. K. Yang, P. Sarkar, C. Weng, and X. Wang. Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In *CCS 2021*, pages 2986–3001. ACM, 2021.
83. R. Yang, M. H. Au, Z. Zhang, Q. Xu, Z. Yu, and W. Whyte. Efficient lattice-based zero-knowledge arguments with standard soundness: Construction and applications. In *CRYPTO 2019*, volume 11692 of *LNCS*, pages 147–175. Springer, 2019.

Author Index

A

Aranha, Diego F. 302
Arriaga, Afonso 3

B

Barbosa, Manuel 3
Bünz, Benedikt 269

C

Chakraborty, Suvradip 101
Chen, Jessica 269
Costache, Anamaria 302

D

Di Giandomenico, Emanuele 134
Di, Zijing 236

G

Garreta, Albert 402
Gu, Yanqi 66
Guimarães, Antonio 302

H

Han, Shuai 34, 168
Hazay, Carmit 367
Heath, David 367

J

Jarecki, Stanislaw 3, 66

K

Kedzior, Pawel 66
Kloöß, Michael 203
Kolesnikov, Vladimir 367

L

Lai, Russell W. F. 203
Lin, Fuchun 337
Liu, Shengli 34, 168
Lyu, You 34

M

Magliocco, Lorenzo 101
Magri, Bernardo 101
Manzur, Ignacio 402

N

Nazarian, Phillip 66
Nguyen, Ngoc Khanh 203
Nguyen, Wilson 236

O

Osadnik, Michał 203
Ouyang, Ying 436

R

Riepel, Doreen 134

S

Schäge, Sven 134
Škrobot, Marjan 3
Soria-Vazquez, Eduardo 302

T

Tang, Deng 436
Tyagi, Nirvan 236

V

Venkitasubramaniam, Muthuramakrishnan
367
Venturi, Daniele 101

W

Wang, Weihao 168

X

Xia, Lucas 236
Xing, Chaoping 337
Xu, Jiayu 66
Xu, Yanhong 436

Y

Yang, Yibin 367
Yao, Yizhou 337